Departamento de Ciências e Tecnologias da Informação

IoPhyR - Physical Rehabilitation IoT System

Sistema de reabilitação motora baseado em andarilhos inteligentes e *IoT*

Carlos Miguel Alpedrinha Ramos de Almeida Nave

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador(a):
Doutor Octavian Postolache, Professor Auxiliar,
ISCTE-IUL

Co-Orientador(a):
Doutor Vitor Viegas, Professor Auxiliar
Escola Naval

Novembro, 2018

Agradecimentos

Primeiramente gostaria de agradecer ao meu Orientador, o Professor Doutor Octavian Postolache e ao meu Co-Orientador, o Professor Doutor Vitor Viegas, pela oportunidade de participar num projecto tão desafiante e inovador. Devo agradecer-lhes também toda a disponibilidade, empenho, ideias, compreensão e motivação dadas no decorrer deste longo projecto.

Em segundo lugar devo agradecer ao projeto *TailorPhy: Smart Sensors and Tailored Environments for Physiotherapy* (PTDC/DTP-DES/6776/2014), financiado pela Fundação para a Ciência e Tecnologia, do qual faz parte este projeto, por proporcionar todos os recursos necessários. Assim como ao Instituto de Telecomunicações e ao ISCTE-IUL pela disponibilização das suas instalações para a investigação e desenvolvimento dum projeto tão exigente e estimulante como o que é objeto desta dissertação.

Finalmente, gostaria de aproveitar esta oportunidade para reconhecer e agradecer aos meus pais, à minha irmã, aos meus amigos próximos e aos meus colegas por toda a disponibilidade, apoio, suporte, e motivação que me foram dados. Sem vocês não teria sido possível concluir este trabalho. Obrigado a todos!

Resumo

A presente dissertação descreve o desenvolvimento de um sistema *IoT* (*Internet of Things*) de reabilitação física baseado em *smart walkers*. O sistema inclui sensores de tipo *IMU* (*Inertial Measurement Unit*), sensores de força (células de carga), e sensores de proximidade (sensores de ultrassons). Os sinais adquiridos por uma plataforma de computação com microcontrolador ligada aos sensores permitem calcular métricas associadas a caracterização da orientação e do equilíbrio do paciente, assim como elementos relacionados com a utilização do andarilho como a elevação do mesmo, o número de passos efetuados e a força exercida sobre os pés do andarilho. O *smart walker* utiliza a plataforma de computação de tipo *Arduino Mega* para a realização do cálculo de métricas ligadas a caracterização das sessões de fisioterapia que serão posteriormente armazenadas na nuvem. Os dados adquiridos ao nível dos *smart walkers* são transmitidos para a nuvem do sistema, utilizando um módulo *Wi-Fi*, ou por intermédio de um *tablet* que recebe os dados da sessão em curso através de comunicação Bluetooth, sendo realizada uma sincronização de dados *tablet-nuvem*.A análise e visualização dos dados armazenados é realizada através da *webapp* e aplicação móvel desenvolvidas.

Palavras-Chave: sensores inteligentes; Reabilitação Física; *IoT*; Sistemas embebidos; Equipamentos de fisioterapia.

Abstract

This dissertation describes the development of an Arduino based physical rehabilitation IoT system. The system uses metrics acquired from IMU sensors (Inertial Measurement Unit), pressure sensors (load cells) and distance sensors (ultrasound sensors). The metrics extracted from these sensors help to determine the patient's orientation, the number of steps taken, and the patient's balance. The smart walker uses the Arduino Mega platform to calculate the required metrics during the physiotherapy sessions to store them in the system cloud server afterwards. The cloud server storing process is done straight from the smart walker, using a Wi-Fi module, or through a mobile device with the system's mobile app, using a Bluetooth module. The stored data analysis and visualization is performed through the developed system's user interfaces (a webapp and a mobile app).

Keywords: Physical; Rehabilitation; IoT; Sensors; Walker; Arduino.

Índice

Índice de Tabelas

Índice de Figuras

Lista de Abreviaturas e Siglas

3G – 3ª Geração

4G – 4ª Geração

ADS – Active Depth Sensor

AFH – Adaptive Frequency Hopping

API – Application Programming Interface

APSD – Automatic Power Save Delivery

AVC – Acidente Vascular Cerebral

BLOB – Binary Large Object

CMOS – Complementar Metal-Oxide Semiconductor

CSR – Cambridge Silicon Radio

DSR – Design Science Research

DSRMPM – Design Science Research Methodology Process Model

ECG – Eletrocardiograma

EDR – Enhanced Data Rate

EEPROM – Electrically Erasable Programmable Read-Only Memory

EER – Extended Entity-Relationship

GPIO – General Purpose Input/Output

HMI – Human Machine Interface

HTTP – Hypertext Transfer Protocol

I2C – Inter-Integrated Circuit

ICFDH – International Classification of Functioning, Disability and Health

ICSP – In-Circuit Serial Programming

IDE – Integrated Development Environment

IMU – Inertial Measurement Unit

IP – Internet Protocol

ITU – International Telecommunication Union

IoE – Internet of Everything

IoP – Internet of People

IoT – Internet of Things

JSON – JavaScript Object Notation

LAMP – Linux, Apache, MySQL, PHP

LIDAR – Light Detection and Ranging

LRF – Laser Range Finder

LTE – Long Term Evolution

MIMO – Multiple Input, Multiple Output

MQTT – Message Queuing Telemetry Transport

MySQL – My Structured Query Language

OMS – Organização Mundial da Saúde

ONU – Organização das Nações Unidas

P2P – Peer to Peer

PHP – PHP: Hypertext Preprocessor

PWM – Pulse Width Modulation

QXGA – Quantum eXtended Graphics Array

RAM – Random Access Memory

RFID – Radio-Frequency Identification

ROM – Read-Only Memory

SDIO – Secure Digital Input Output

SDK – Software Development Kit

SO – Sistema Operativo

SPI – Serial Peripherical Interface

SPP – Serial Port Protocol

SQL – Structured Query Language

SQLite – Structured Query Language Lite

SRAM – Static Random Access Memory

SSH – Secure Shell

STBC – Space-Time Block Codes

TCP – Trasmission Control Protocol

UART – Universal Asynchronous Receiver-Transmitter

URL – Uniform Resource Locator

USB – Universal Serial Bus

VPS – Virtual Private Server

VoIP – Voice over Internet Protocol

WSN – Wireless Sensor Networks

XML – Extensible Markup Language

Capítulo 1 – Introdução

Neste capítulo procura-se realizar uma contextualização para a temática da investigação explanando para isso o seu enquadramento, motivação, questões, objectivos e métodos.

1.1. Enquadramento do tema

A fisioterapia, executada em forma de sessões e de forma periódica, atua como uma parte crucial no restabelecimento das capacidades físicas dos pacientes. Estudos reportam os beneficios da utilização de equipamentos de apoio à marcha como muletas ou andarilhos nas sessões de fisioterapia [1], embora estes equipamentos não forneçam informação objectiva sobre o processo de fisioterapia.

As sessões de reabilitação são supervisionadas por um ou mais fisioterapeutas que acompanham o paciente durante a mesma, certificando-se que este realiza com sucesso as atividades planeadas para essa sessão. No entanto existem métricas difíceis de analisar visualmente pelos profissionais que tornam subjectiva a avaliação do progresso do paciente. Uma avaliação objectiva só poderá ser obtida através da utilização de equipamentos de reabilitação com sensores inteligentes que permitem recolher sinais e calcular métricas que caracterizam o processo de reabilitação.

Os equipamentos "*Smart*", na fisioterapia (andarilhos, muletas etc..), fornecem informações sobre o processo de reabilitação, através de uma análise objetiva baseada no cálculo de métricas difíceis de ver a olho nu [2]. Estas métricas são rapidamente calculadas, e disponibilizadas aos fisioterapeutas, de forma estruturada, para que estes posam ter a perceção da situação clínica do paciente, podendo ser assim tomadas as decisões mais indicadas de forma a atinjir uma melhor recuperação de uma forma mais célere.

Têm-se desenvolvido ainda sistemas equipados com motores que incorporam interfaces *HMI* (*Human Machine Interfaces*), sensores do tipo *IMU* e *LRF* de maneira a detectar parâmetros da marcha e facilitar os movimentos dos pacientes, tornando mais segura a realização de sessões de fisioterapia [3].

1.2. Motivação e relevância do tema

A fisioterapia é uma atividade e uma ciência que se dedica à recuperação da mobilidade dos pacientes mediante a execução de exercícios que estimulem a musculatura, com ou sem o uso de aparelhos. As razões que levam a que a população recorra a este tipo de serviços tem que ver com a existência de eventos de tipo AVC, doenças degenerativas ou outras doenças que limitem a mobilidade, como acidentes que possam ter sofrido ou inclusive com a redução da mobilidade associada ao avançar da idade.

Uma das principais limitações das sessões de fisioterapia prende-se com o facto dos aparelhos de fisioterapia tradicionais (ex: andarilhos e muletas), por si sós, não permitirem uma análise objectiva da evolução dos pacientes [4].

O desenvolvimento do tipo de plataformas como a que foi desenvolvida nesta dissertação tem como motivação e objetivo conseguir obter uma análise mais minuciosa e rápida do estado de recuperação do paciente. Este tipo de sistema é benéfico na recuperação de problemas motores independentemente da faixa etária do paciente ou da origem da sua limitação motora. No entanto tem um maior impacto na recuperação da 3ª idade, já que lhes garante a estabilidade e confiança necessárias para realizarem as sessões de fisioterapia sem o receio de terem quedas, que se tornam comuns com o avançar da idade [5], [6].

É do conhecimento geral que a população mundial está a envelhecer devido aos avanços na medicina e à cada vez mais baixa taxa de natalidade. Contudo, um estudo das Nações Unidas de 2015 vem agravar as espectativas em relação a este fenómeno demográfico, salientando que o número de pessoas com idade superior aos 60 anos chegará, em 2050, ao dobro dos valores actuais [7]. Este factor, em conjunto com a existência de incapacidades decorrentes do avançar da idade (ex: falta de equilíbrio ou força muscular) impulsionam a investigação e desenvolvimento nesta área, uma vez que se pretende ter uma atuação não só num evento específico, como também que este setor crescente da população consiga manter o maior grau possível de autonomia. A recuperação da referida autonomia e do equilíbrio previne quedas, que em indivíduos idosos ou com incapacidades podem reduzir bastante a sua qualidade de vida, podendo mesmo revelar-se situações críticas [5], [8], [9].

Este tipo de plataformas é importante não só para os indivíduos que são afetados diretamentente por estas melhorias nos sistemas de fisioterapia, como também para a população em geral devido ao fenómeno demográfico descrito anteriormente.

1.3. Objetivos de investigação

A presente dissertação tem como propósito o desenvolvimento de um sistema de reabilitação motora compatível com *IoT*. Os objectivos principais associados a presente investigação científica são:

- Desenvolvimento de um andarilho inteligente, ou *smart walker*, como elemento principal do sistema, que auxilie os fisioterapeutas na avaliação da performance e da evolução dos pacientes durante os planos de tratamento.
- Desenvolvimento de algoritmos para o cálculo da orientação do paciente utilizador do *smart walker*, da medição de elevação do andarilho, do cálculo do número de passos realizados, da força aplicada nos pés do andarilho e do equilíbrio do paciente durante os treinos de marcha apoiada pelo andarilho.
- Desenvolvimento de um módulo de identificação (*RFID*), com a finalidade de registar e identificar os utilizadores de andarilho através de utilização de cartões que lhes são atribuídos.
- Desenvolvimento do servidor do sistema, o *PhysioWatch*, para a gravação e análise dos dados das sessões de fisioterapia, assim como para interligação com as aplicações do sistema.
- Desenvolvimento de uma aplicação móvel que permita aceder aos dados guardados no servidor, e realizar sessões em *real-time*.

1.4. Questões de investigação

O sistema desenvolvido pretende prover o fisioterapeuta de dados que permitam obter uma análise objectiva das incapacidades e da evolução clínica dos pacientes, de modo a adaptar os tratamentos e auxilar o processo terapêutico. Nesse sentido levantaram-se as seguintes questões de investigação que se pretendem responder nesta dissertação:

- Quais são os equipamentos mais utilizados nas clinicas de fisioterapia para reabilitação da marcha?
- Quais são as métricas indicadas para obter a avaliação de pacientes durante as sessões de fisioterapia?

- Quais são os sensores que podem fornecer a informação sobre o processo de reabilitação a ser realizado com o equipamento de fisioterapia?
- Qual é a forma mais indicada para visualização dos dados das sessões por parte dos fisioterapeutas?

1.5. Abordagem metodológica

O *Design Science Research* (*DSR*) descreve uma metodologia de investigação que tem como objectivo a solução de problemas práticos ou teóricos recorrendo à criação de inovadores artefactos de tecnologias de informação [10]. Foi escolhida esta metodologia já que se trata de um problema da "vida real" e dado que a sua natureza é prática, abstrata e humana.

O *Design Science Research Methodology* é constituído por 4 pontos de entrada, ou abordagens, consoante o estado do artefacto no momento em que se inicia a investigação [10]:

- Abordagem centrada no problema.
- Abordagem centrada no objetivo.
- Abordagem centrada no design e desenvolvimento.
- Abordagem contextualizada.

Uma vez que a ideia de investigação foi obtida através da observação de um problema, adotou-se a primeira abordagem.

Esta metodologia tem um *Design Research Methodology Process Model* (DSRMPM) que é composto por seis fases ou atividades [10]:

1. **Motivação e identificação do problema:** Ambos descritos nas secções de Motivação e Questões de investigação, respetivamente.
2. **Definição de objetivos:** Os objetivos estão amplamente descritos na secção objetivos de investigação. No entanto o principal objetivo é o de desenvolver uma solução de reabilitação física e uma aplicação móvel que permita a monitorização em tempo real.
3. **Projeto e desenvolvimento:** Nesta fase é realizada a definição dos componentes a utilizar (placas de desenvolvimento, sensores e atuadores), os aparelhos de fisioterapia onde os acoplar (muletas, andarilhos e afins), as métricas a recolher

e a arquitetura de rede a adotar. Após a definição é realizado então o desenvolvimento, onde se inclui a extração das métricas e o seu devido armazenamento numa base de dados e o desenvolvimento da referida arquitetura de rede.

4. **Demonstração:** Nesta etapa são realizados ensaios e simulações com voluntários de maneira a calibrar e perceber as melhoras que o sistema traz a pacientes e profissionais da fisioterapia.
5. **Avaliação:** Nesta fase pretende-se perceber se os objetivos determinados no início para a dissertação foram atingidos, mediante uma comparação com as expetativas teóricas com os resultados práticos obtidos.
6. **Comunicação:** Nesta última fase espera-se comunicar o problema que originou a elaboração da dissertação e a importância do seu estudo. Pretende-se ainda comunicar a solução proposta e os seus resultados através da elaboração de um artigo científico.

1.6. Estrutura e organização da dissertação

A presente dissertação está organizada em oito capítulos que pretendem refletir as diferentes fases do desenvolvimento da mesma até a sua conclusão.

- O primeiro capítulo introduz o tema da investigação e objetivos da mesma bem como uma breve descrição da estrutura do trabalho.
- O segundo capítulo reflete o enquadramento teórico, designado por Revisão da literatura.
- O terceiro capítulo é dedicado a uma descrição abrangente do sistema desenvolvido na presente dissertação, assim como as suas funcionalidades e componentes de *hardware*.
- O quarto capítulo apresenta apresenta o protótipo de andarilho desenvolvido, incluindo os seus componentes de *hardware* e o *software* embebido criados para ele.
- O quinto capítulo é dedicado à descrição do servidor do sistema, incluindo os seus conteúdos e as interações criadas.
- O sexto capítulo apresenta as aplicações criadas para o utilizador do sistema, onde os utilizadores podem realizar as acções que lhes são permitidas.

- O sétimo capítulo é dedicado à apresentação de resultados obtidos nos ensaios realizados com o sistema, assim como a análise dos mesmos.
- O oitavo capítulo apresenta as conclusões obtidas com o término do desenvolvimento do sistema assim como o trabalho a ser realizado futuramente.
- Finalmente, são disponibilizados nos apêndices os artigos científicos realizados, e os manuais do utilizador e técnico que pretendem explicar as funcionalidades das aplicações do sistema desenvolvido, assim como as recomendações de utilização.

Capítulo 2 – Revisão da Literatura

Neste capítulo apresentam-se os pontos chave de investigação utilizados na elaboração da presente dissertação.

2.1. O contexto da reabilitação física

Nesta secção pretende-se apresentar o paradigma da reabilitação física, a sua evolução, os problemas que enfrenta e as soluções para esses mesmos problemas.

2.1.1. Reabilitação física

A reabilitação física e motora é parte fundamental na recuperação dos pacientes independentemente da sua limitação ser resultado de um acidente, duma doença ou do avançar da idade [11]. Sendo que o uso de aparelhos como andarilhos ou muletas não só não interferem nos resultados obtidos como em alguns casos reduzem o período de reabilitação [1], [12].

Os fisioterapeutas exercem a sua profissão através da reabilitação física de pacientes com algum tipo de incapacidade física. A reabilitação é um processo através do qual se espera que o paceinte aumente a sua qualidade de vida, recuperando ou melhorando a sua condição física, conseguindo ao mesmo tempo reaver a sua autonomia e independência de maneira a poder retornar à vida ativa, tanto social como laboral, podendo participar na sociedade e economia com a sua independência económica [13].

Dentro dos equipamentos utilizados pelos fisioterapeutas para auxiliar a recuperação dos pacientes, encontram-se as muletas e andarilhos que permitem manter o equilíbrio e a posição ao mesmo tempo que se realiza a sessão de fisioterapia [14].

2.1.2. O problema

Segundo um relatório da OMS, cerca de 15% da população mundial sofre com algum tipo de incapacidade motora. Sendo que destes, 4% sofrem com incapacidades extremamente severas e limitadoras (ex: tetraplegia ou cegueira). As incapacidades são definidas pelo *ICFDH* e pela OMS como um termo mais lato para restrições de actividade e de participação do indivíduo na realização de tarefas ou trabalhos específicos, e/ou em situações do dia-a-dia [13].

O aparecimento de incapacidades, segundo a OMS, tem como causas a falta de bons cuidados médicos, com comportamentos pouco saudáveis (ex: má alimentação ou toma de estupefacientes), com o surgimento de doenças que as despelotam, ou com o

aumento da esperança de vida. Sendo que esta última está relacionada com o aumento de probabilidades de aquisição de incapacidades com o avançar da idade, através do aparecimento de doenças crónicas ligadas ao sistema motor [13].

Esta última causa é preocupante na medida em que se espera que devido ao avanço da medicina nas últimas décadas, que produziu um aumento da esperança de vida, a população mundial com idade superior a 60 anos, e que potencialmente necessite deste tipo de tratamentos, supere o dobro dos valores actuais em 2050, isto é 2.1 biliões [7].

2.1.3. A solução

A OMS sugere um modelo de ação e prevenção de incapacidades motoras que inclui três fases. A primeira visa a diminuição das incapacidades executando para isso medidas como a educação. A segunda por outro lado, tem como objetivo o rastreio e identificação das incapacidades num estado ainda embrionário, conseguindo-se mitigar ou eliminar o problema. Na terceira e última fase aparece um dos temas desta dissertação, a reabilitação física, como ferramenta que possibilita a redução de problemas associados e a recuperação da autonomia [15].

A organização referida, a OMS, sublinha ainda a importância da reabilitação física como forma de atingir o objetivo de desenvolvimento sustentável, face à existência de um défice de práticas de reabilitação a nível mundial, devido ao aumento de população com este tipo de necessidades. Neste relatório é referido ainda, um plano de ação mundial sobre a incapacidade e a promoção de uma conferência intitulada "*Rehabilitation 2030*", realizada no início de 2017, onde se discutiu com os estados membros, representantes e provedores de serviços desta área da saúde, uma ação coordenada com o fim de melhorar o sector e faze-lo chegar às pessoas que dele necessitem [11] .

Noutro relatório sobre as incapacidades, a OMS estabelece que a reabilitação física está inacessível a uma larga franja da população mundial. Devendo-se este facto a vários factores, sendo os principais a falta global de profissionais da área, a não incorporação deste tipo de serviços/tratamentos na legislação para a saúde, a não divulgação e promoção dos seus benefícios, o não incentivo generalizado dos governos à criação deste tipo de serviços, e ao custo elevado deste tipo de tratamentos tendo em conta a economia local. Nesse sentido, foi apresentada como solução o incentivo à criação e acesso de dispositivos tecnológicos de assistência para os pacientes, de maneira a atingir

uma maior participação e redução dos custos da saúde, não eliminando, no entanto, o contacto com o profissional de saúde, de maneira a que este possa monotorizar a recuperação [13].

2.2. A tecnologia na reabilitação física

Nesta secção pretende-se demonstrar a necessidade do desenvolvimento de sistemas que permitam uma análise objetiva dos tratamentos de fisioterapia, assim como apresentar a sua aplicação.

2.2.1. Aumento da investigação na área da reabilitação física

O aumento da investigação na área da reabilitação física vem sustentado na necessidade de haver uma análise objetiva da evolução dos pacientes, na falta de profissionais na área da fisioterapia, e no elevado preço das sessões de fisioterapia, como descrito num relatório da OMS já referido [11], [16].

2.2.2. Sistemas de IoT na reabilitação física

Estes são sistemas centrados no paciente que realizam a monitorização em tempo real da condição física do paciente, permitindo que as sessões se realizem de forma presencial ou remota. Estes sistemas possibilitam também a interoperabilidade entre o paciente e o profissional de saúde, o diagnóstico automático, de forma eficiente e não obstrutiva, permitindo que a sessão de reabilitação física flua de forma natural, progressiva e sem paragens [17].

A referida monitorização em tempo real, a interoperabilidade entre o paciente e fisioterapeutas, e a possibilidade de correção de execução de exercícios de imediato, melhoram a comunicação paciente-fisioterapeuta, podendo-se assim atingir melhores e mais rápidos resultados, diminuindo assim o tempo de recuperação e por consequência os custos associados ao tratamento [1], [18].

Os sistemas desenvolvidos para a fisioterapia utilizam equipamentos comuns nos tratamentos, como andarilhos, onde se aplicam sistemas embebidos com sensores para calcular métricas complexas. Métricas essas que são guardadas em bases de dados locais ou servidores ficando disponíveis para os profissionais, de maneira a auxiliá-los a realizar diagnósticos, tornando os tratamentos mais rápidos e eficazes [2].

Artigos relatam o desenvolvimento de aparelhos inteligentes como *smart walkers*, *smart crutches*, *wearables*, plataformas de força ou esferas para a realização da reabilitação física de pacientes. Estes aparelhos, utilizam *IMU*, acelerómetros, magnetómetros ou giroscópios, sensores de pressão, de força ou sensores de movimento Radares *doppler*. Com estes sensores pretendem determinar o equilíbrio, a marcha, a posição, a orientação ou as forças aplicadas pelo paciente. Alguns destes sistemas desenvolvidos incluem ainda aplicações móveis onde se regista e se disponibiliza os dados recolhidos das sessões de fisioterapia [19]–[24]. Existem ainda outros artigos que revelam a aplicação de tecnologias interativas como a realidade virtual e sistemas como o *Kinnect* da *Microsoft*, que realizam a deteção remota de movimentos com tecnologia óptica, para extraírem os dados biométricos dos pacientes de forma lúdica [25]–[28].

2.2.3. Aplicação de sistemas de reabilitação física em andarilhos

Existe um artigo que refere o desenvolvimento de um sistema de reabilitação física, aplicado em andarilhos que usam sensores *ADS* e algoritmos de deteção de padrões de marcha e da posição dos pés do paciente, para melhorar a sua condição física [29]. Num outro artigo, apresenta-se o desenvolvimento de um *smart walker* que utiliza um *IMU* e sensores de força para auxiliar a recuperação de pacientes [2]. Outros autores apresentam *smart walkers* e algoritmos que analisam os padrões de marcha, e a distância entre pés para detectar a perda de equilíbrio de maneira a prevenir quedas [30]. Outro artigo refere a utilização de sensores *LIDAR* e de motores vibratórios em andarilhos para ajudar pessoas cegas a andar em ambientes complexos, evitando a colisão com obstáculos [31].

2.3. A *Internet of Things* e a *Internet of Everything*

Nesta secção pretende-se apresentar os conceitos da *Internet of Things* (IoT), e da *Internet of Everything* (IoE) no contexto de aplicações para a saúde.

2.3.1. *Internet of Everything*

Este conceito foi primeiramente introduzido por Dave Evans, *Chief of Futurism* na *Cisco*, em 2012. (Figura 2.1). O IoE é mais abrangente que o da IoT, já que resulta da fusão deste com o da IoP (*Internet of People*), ligando as pessoas, as coisas os processos e os dados [32], [33].

A *IoT* é definido como um grupo de infraestruturas que interligam "coisas" entre si, permitindo a sua gestão e o acesso aos dados por eles gerados [34] . Já a *IoP*, é um

conceito gerado pela alteração da posição do ser humano em relação à tecnologia. Isto é, com a evolução da tecnologia e o surgimento de *werables*, dispositivos inteligentes como os *smart watches*, o ser humano passou a ser parte da rede e já não só um mero observador, devido à sua interação mais natural com este tipo de dispositivos [33], [35].

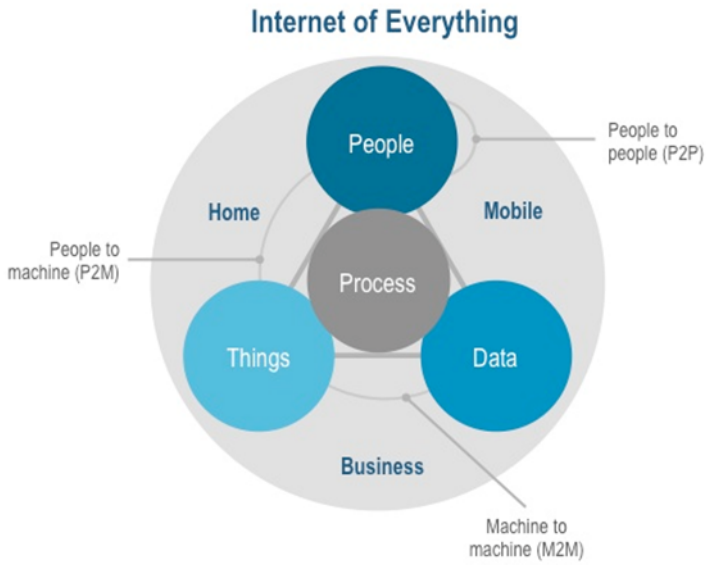


Figura 2.1 – Esquema da *Internet of Everything*.

A *IoT* ao ligar as pessoas aos outros três pilares, através de dispositivos *werables*, *smartphones*, *tablets* e outros sem acesso à Internet, gera um enorme volume de dados. Dados esses que ao serem extraídos, processados e analisados em tempo real, com ferramentas *Big Data*, permitem cumprir metas governativas relativas à qualidade de vida da população [36], [37].

2.3.2. *Internet of Things*

O rápido desenvolviemento de sensores na última década tornou possível a disseminação e implementação do conceito *IoT*. Este conceito foi proferido primeiramente em 1999 pelo Kevin Ashton, um dos fundadores do *Massachusetts Insititute of Technology Auto-ID Center* [38], no entanto só se tornou mais conhecido quando em 2005 o *ITU* publicou o primeiro relatório sobre o assunto. No relatório apresentava-se o *IoT* como sendo uma tecnologia aglomeradora que ligaria os sensores e as suas redes, *WSNs*, e os sistemas embebidos, sendo os sensores os "*feeling things*", ou leitores de métricas, e os sistemas embebidos os "*thinking things*", ou unidades de processamento dessas métricas [39].

Mas recentemente o *IoT* é descrito como uma rede inteligente que interliga todos os dispositivos à internet, e que realiza a comunicação e transporte dos dados recolhidos pelos nós sensores dentro da rede, de acordo com os protocolos escolhidos e tem como objetivos a monitorização, controlo, localização, a identificação e a atuação conforme os dados recolhidos [40].

Hoje em dia temos uma rede que interliga os computadores e que conta com cerca de 1.5 mil milhões de computadores entre si, a internet, no entanto projeta-se que este número venha a aumentar entre 30 a 60 vezes até 2022. Este esperado aumento está relacionado com o surgimento e a massificação do conceito da *IoT*, através da criação de sistemas *IoT* com nós interligados entre si através da internet [41].

O surgimento da *IoT* vem trazer a possibilidade de criação de soluções para muitos problemas ou ineficiências da sociedade, como as mostradas na Figura 2.2, conseguindo melhorar dessa maneira a qualidade de vida dos seus integrantes [42]. Dentro dos sectores onde se pensa utilizar ou já se utiliza este conceito, a saúde apresenta-se como sendo uma área muito atrativa, tanto que é esperado que até 2020, 40% das soluções *IoT* sejam deste ramo, tornando-se num mercado avaliado em 117 mil milhões de dólares. Nesta área os principais objetivos são os de construir soluções mais baratas, que tenham um bom rácio custo-eficiência, que melhorem a qualidade de vida dos pacientes e que consigam atingir melhores resultados [43].

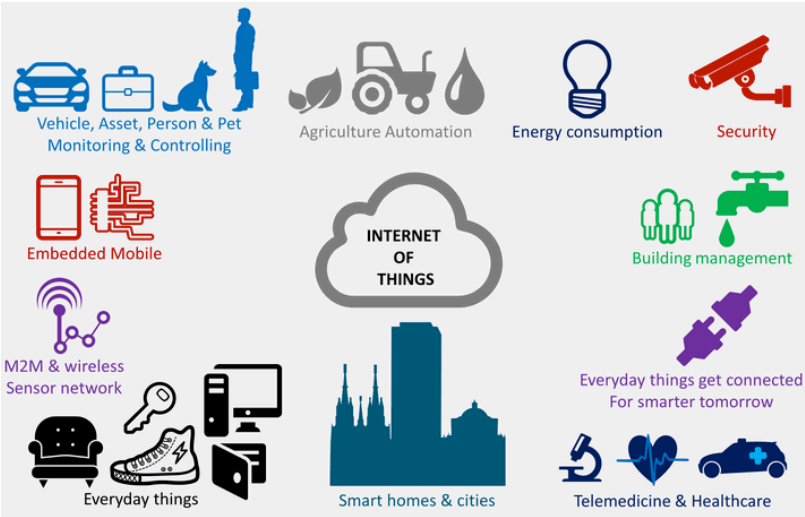


Figura 2.2 – Esquema da *Internet of Things*.

2.4. As aplicações móveis e o sector da reabilitação

Desde 2011 existem dois sistemas operativos móveis que dominam o mercado de consumo de *smartphones* e *tablets*. Estes são o *iOS* e o *Android* [44]. O primeiro é um SO proprietário desenvolvido pela *Apple*, que só corre nos seus dispositivos, tendo um ecossistema fechado e pouco personalizável. O segundo, por outro lado, é um SO *open source* desenvolvido pela *Google* presente em *tablets* e *smartphones* de diversas marcas, que é baseado no *kernel* do *Linux* e que é flexível e reprogramável [45],[46].

Em relação à utilização, o *Android* é o SO mais utilizado, tendo atingido no 2º trimestre de 2018 uma quota de mercado de 88% dos dispositivos móveis [44]. Entre as razões para esta predominância, destaca-se o facto de ter sido o primeiro SO móvel *open source*. O que fez com que houvesse interesse dos desenvolvedores e fabricantes, já que era gratuito e personalizável. Outra razão, que é consequência da primeira, é o facto de ter chegado a diversos fabricantes e a todas as gamas de preços de dispositivos, o que torna a sua aquisição comportável a quase toda a população [47].

O facto do *Android* ter mais fabricantes e utilizadores a utilizá-lo, fez com que houvesse um mercado maior para os desenvolvedores explorarem. Nesse sentido, foram criadas aplicações para diversos fins e sectores, tendo-se atingido no 1º trimestre de 2017 a marca 3.2 milhões de aplicações desenvolvidas para este sistema. O que corresponde a quase o dobro das aplicações desenvolvidas para o *iOS* em igual período [48]. Entre os sectores objeto de desenvolvimento, destaca-se o da saúde, para o qual têm sido desenvolvidas inúmeras aplicações para auxiliar a decisão dos profissionais com base em dados dos pacientes [49]. Seguindo essa tendência, em 2017, foram desenvolvidas aplicações para o *Android OS* que integram sistemas de reabilitação física. Estas aplicações permitem que os seus utilizadores acedam aos dados registados durante sessões de fisioterapia de maneira visualizá-los e ajudar os fisioterapeutas na criação de diagnóstico dos seus pacientes [25], [50], [51].

Capítulo 3 – Descrição do sistema

Neste capítulo pretende-se descrever todo o sistema desenvolvido na presente dissertação, dando enfase à arquitetura da rede, funcionamento e componentes de *hardware*.

3.1. Arquitetura do sistema

A arquitetura do sistema desenvolvido na presente dissertação é baseada na plataforma de computação *Arduino* e é dividida em 3 partes, sendo a primeira o objecto inteligente (nó da rede), o andarilho, ou *smart walker*, a segunda o servidor, e a terceira um dispositivo com a aplicação móvel do sistema, a *PhysioApp* (Ver Figura 3.1).

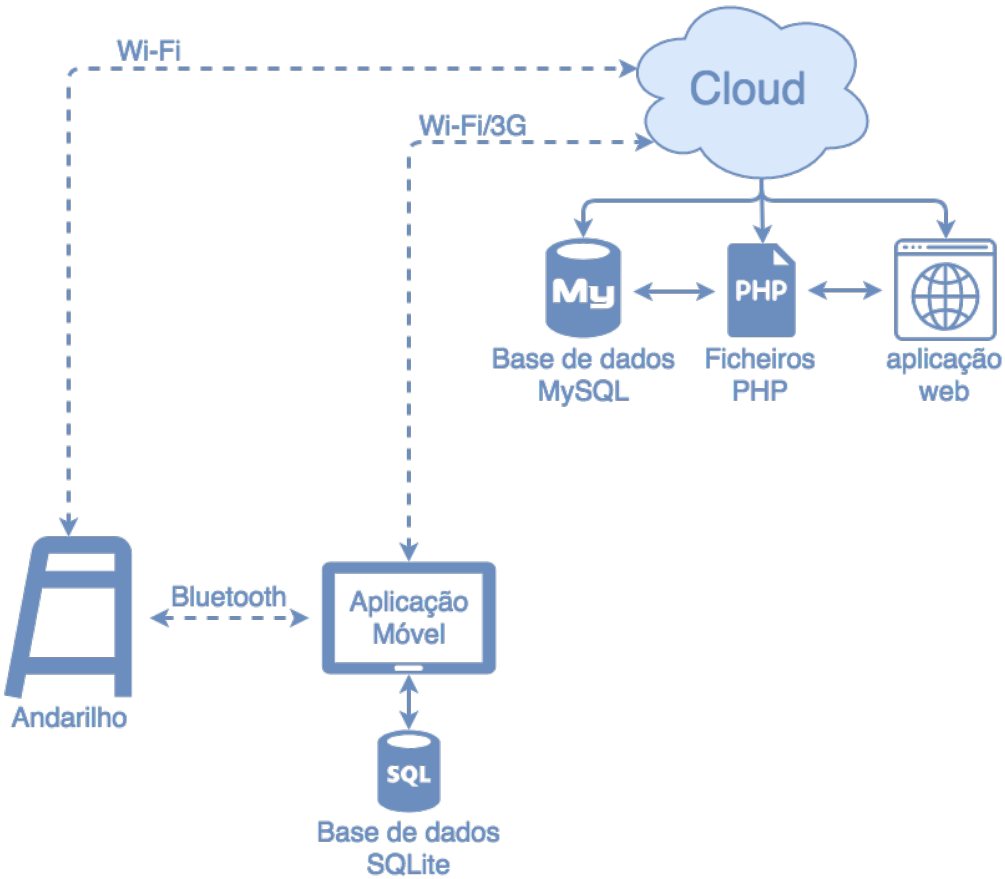


Figura 3.1 – Diagrama da arquitetura do sistema.

A primeira parte da arquitetura de rede é o andarilho inteligente, onde está colocada uma plataforma de desenvolvimento *Arduino Mega* que realiza a aquisição dos sinais analógicos dos sensores ligados às entradas analógicas desta. Os sinais adquiridos são usados para calcular localmente as métricas ligadas ao processo de fisioterapia sendo enviadas posteriormente para o servidor do sistema, na nuvem. A transmissão de informação é realizada utilizando o protocolo de comunicação *Wi-Fi* ou *Bluetooth*.

Utilizando o primeiro, a comunicação é establecida com o servidor do sistema e os dados são armazenados na base de dados do mesmo. Já utilizando o segundo, a informação é enviada para um dispositivo com a aplicação móvel do sistema,a *PhysioApp*.

A segunda parte é a *PhysioApp*, aplicação móvel do sistema, que após receber as métricas enviadas pela plataforma de computação do andarilho, processa-as e mostra-as em tempo real em gráficos adequados para o efeito. A aplicação tem botões para controlar o início e o fim da sessão, e é nela que se identifica qual o paciente que está a realizar os exercícios. Após premir o botão para terminar a sessão, a aplicação guarda os dados da sessão decorrida e envia-os para o servidor, actualizando a base de dados. Para além do referido, a aplicação também acede a informações alojadas no servidor, como dados pessoais, histórico de sessões de fisiotereapia ou planos de exercícios, de maneira a apresentá-las aos seus utilizadores.

A terceira e última parte da arquitetura é o servidor do sistema, o *PhysioWatch*, onde são armazenados os dados. O servidor contém uma base de dados *MySQL*, onde estão presentes todos os dados dos utilizadores e das sessões de fisioterapia, uma aplicação web, a *PhysioWebapp*, onde os utilizadores podem consultar os dados aos quais têm acesso e vários ficheiros *PHP* de modo a realizar a comunicação e troca de dados com a aplicação móvel ou com a aplicação *web* previamente citados.

3.2. Utilizadores e aplicações

O sistema desenvolvido contempla 3 perfis de utilizadores. O administrador, o fisioterapeuta e o paciente. Todos os utilizadores tem acesso ao sistema através da *PhysioWebapp* ou da *PhysioApp*, sendo que o paciente utiliza ainda a aplicação embebida através da realização de sessões de fisioterapia utilizando o andarilho inteligente desenvolvido.

Os utilizadores do sistema diferenciam-se ainda pelo seu papel e pelas funcionalidades que este lhes permite realizar, como se pode ver na Figura 3.2.

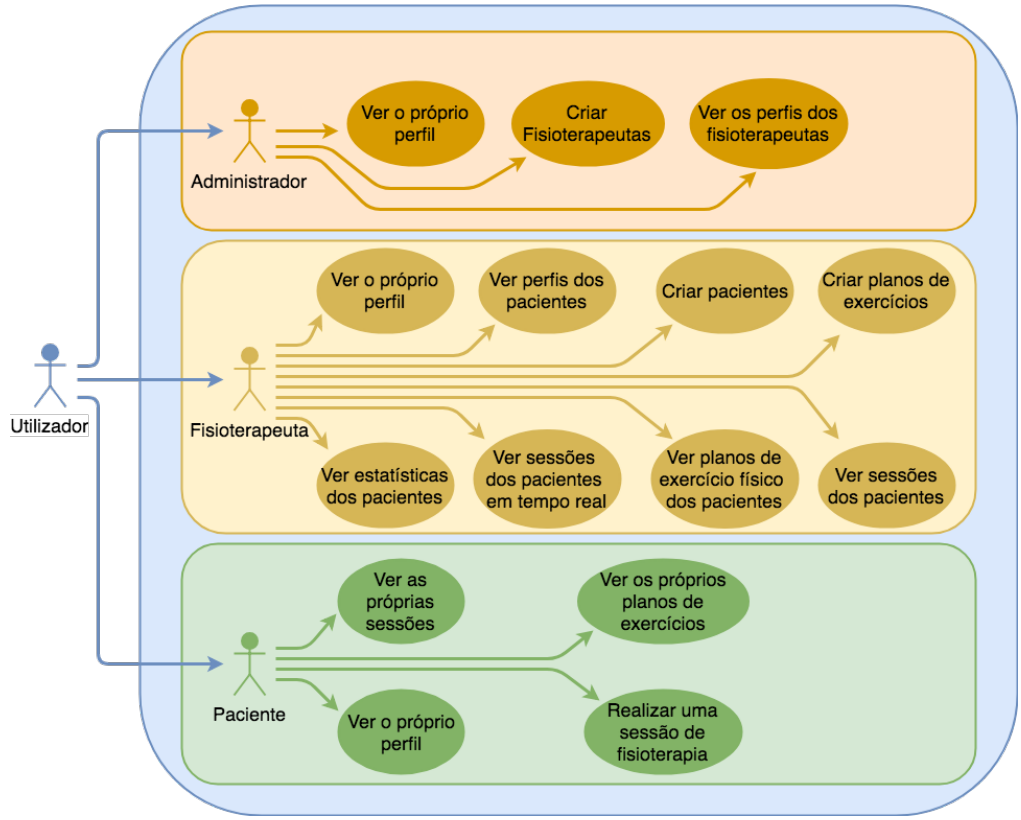


Figura 3.2 – Diagrama de *Use Cases* das funcionalides do sistema.

Em relação às aplicações, foram desenvolvidas três aplicações diferentes que integram o sistema:

- **Aplicação embebida:** Desenvolvida em *C*, para correr na plataforma de desenvolvimento *Arduino* que está colocada no andarilho. Esta aplicação é responsável pelo cálculo das métricas necessárias e o seu envio tanto para o servidor como para os dispositivos móveis com a aplicação móvel do sistema. A aplicação está amplamente descrita no Capítulo 4.
- ***PhysioApp*:** Aplicação móvel desenvolvida em *Java* para o sistema operativo móvel *Android*, que permite aos utilizadores realizarem diferentes acções dependendo do tipo de funcionalidades que lhes são permitidas, como se pode ver na Figura 3.2. Esta aplicação é aplamente descrita no Capítulo 6.
- ***PhysioWebapp*:** uma aplicação *web* que fornece as mesmas funcionalidades que a aplicação móvel do sistema, com a excepção da visualização de dados das sessões de fisioterapia em curso, e das estatísticas do paciente. Esta aplicação está minunciosamente descrita no Capítulo 6.

3.3. Sistema *IoPhyR*

A Figura 3.3 apresenta o fluxograma da utilização do sistema *IoPhyR* por parte dos seus utilizadores, sendo que as letras presentes entre parenteses representam as etapas cronológicas que tem de ser executadas para o funcionamento do mesmo.

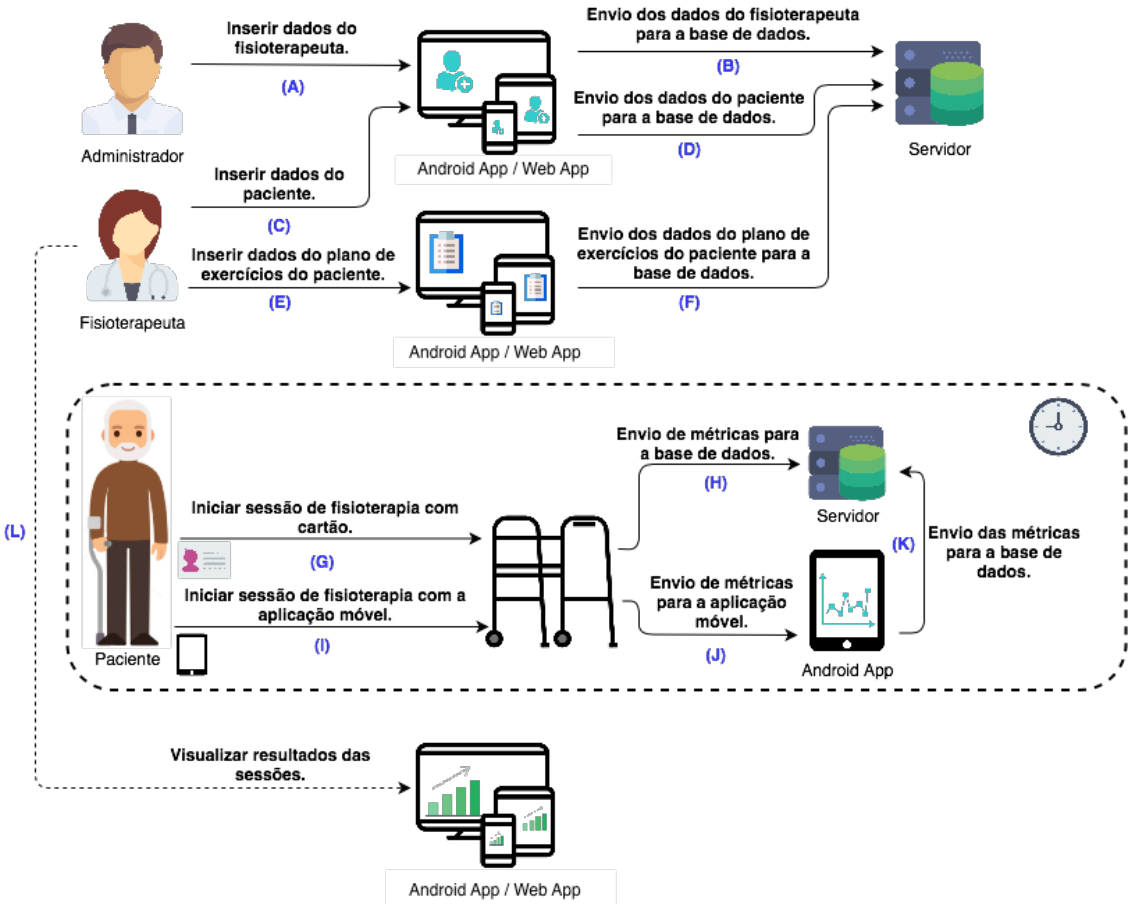


Figura 3.3 – Fluxograma do sistema *IoPhyR*.

Primeiramente, deve ser criado um fisioterapeuta. Para isso o administrador do sistema, o único com direitos para tal, deve preencher um formulário com os dados do fisioterapeuta usando a PhysioApp ou a *PhysioWebapp* **(A)**. Após o preenchimento desse formulário, a aplicação móvel ou a *web* iniciam um pedido *HTTP* através do ficheiro *PHP* alojado no servidor, que comunica com a base de dados *MySQL* do sistema para inserir os dados do novo fisioterapeuta. De seguida o administrador recebe uma resposta *HTTP* com uma mensagem que refere se o registo foi ou não bem sucedido **(B)**.

Após a criação do fisioterapeuta, e para que este possa analisar os dados de pacientes, um paciente deve ser criado. Para isso o fisioterapeuta, o único tipo de utilizador que o

pode fazer, deve preencher um formulário com os dados do paciente, podendo faze-lo utilizando um dispositivo com a *PhysioApp* ou acedendo à *PhysioWebapp*, aplicação *web* do sistema **(C)**.

Entre os dados pedidos no formulário, encontra-se o *ID* do paciente, que é o número de identificação guardado numa *tag RFID*, como as apresentadas na Figura 3.4. A tag RFID é fornecido ao paciente no acto do registo. Esse *ID* é lido por pelo *PhysioRegister*, que inclui um *leitor RFID com capacidades de comunicação Bluetooth,* e é transmitido para a *PhysioApp.*

Figura 3.4 – Cartão e *tag RFID.*

Após o preenchimento do formulário, e à semelhança do que acontece com a criação dos fisioterapeutas, a *PhysioApp* ou a *PhysioWebapp* tentam inserir os dados do paciente na base de dados do sistema, executando para isso um pedido *HTTP* através do ficheiro *PHP* criado para o efeito que está alojado no *PhysioWatch*, servidor do sistema. Finalmente, o fisioterapeuta irá receber uma resposta *HTTP* ao seu pedido referindo se o registo foi ou não bem sucedido **(D)**.

Posteriormente, o fisioterapeuta deve criar um plano de exercícios para o seu paciente. Para isso deve utilizar a *PhysioWebApp* ou a *PhysioApp* para preencher o formulário criado para o efeito. O formulário tem uma secção relativa ao plano de exercícios como um todo, e outra relativa aos exercícios nele incluídos **(E)**.

A primeira secção inclui os seguintes parâmetros que devem ser preenchidos:

- **Data do plano de exercícios:** Esta é data na qual o paciente irá realizar o plano de exercícios. No entanto, o plano criado poderá voltar a ser realizado se o fisioterapeuta assim o entender.

- **Duração do plano de exercícios:** Esta duração, em minutos, representa o tempo da plano de fisioterapia, que o fisioterapeuta deverá distribuir pelos exercícios que o contêm.
- **Nível de dificuldade do plano de exercícios:** Este é um valor no intervalo [1-10], que o fisioterapeuta deverá selecionar em função do exercícios que estão incluídos no plano.

A segunda secção permite ao fisioterapeuta criar múltiplos exercícios, e contém os seguintes parâmetros que devem se preenchidos pelo mesmo:

- **Imagem do exercício:** Esta é uma imagem que deve representar o movimento desejado para o exercício em questão.
- **Título do exercício:** Este é o título do exercício. O título deve ser curto e identificativo.
- **Número de repetições do exercício:** Este é um número no intervalo [0-100], que representa o número de repetições do exercício que o paciente deve realizar.
- **Duração do exercício:** Esta é a duração em minutos que o exercício deverá ter.
- **Nível de dificuldade do exercício:** Este é um número definido pelo fisioterapeuta consoante a dificuldade demonstrada pelo paciente na execução de um exercício. Este valor está compreendido entre 0 e 10, sendo atribuído 0 quando o paciente não tem qualquer dificuldade, e 10, quando este não consegue realizar o exercício.
- **Descrição do exercício:** A descrição do exercício que o paciente deve realizar. Esta está limitada a 200 caracteres de maneira a ser objetiva e sintética, complementando a informação visual e textual, da imagem e título do exercício.

De seguida, e após o preenchimento do formulário do plano de exercícios, a *PhysioApp* a *PhysioWebapp*, envia um pedido *HTTP* através do ficheiro *PHP* criado para o efeito, que tenta inserir o novo plano de exercícios na base de dados *MySQL* do sistema. Seguidamente, o fisioterapeuta recebe uma resposta *HTTP* com uma mensagem que indica se o registo foi ou não bem sucedido **(F)**.

Estando já criado o paciente, e atribuído um plano de exercícios, este poderá realizar uma sessão de fisioterapia. Esta poderá ser iniciada de 2 formas diferentes. Através do cartão *RFID* atribuído ao paciente, ou através da *PhysioApp*.

No primeiro caso, o paciente, ou o seu fisioterapeuta, terá de encostar o cartão *RFID* do paciente ao leitor de cartões *RFID* presente no andarilho. Caso o paciente esteja registado um *LED* verde acenderá ao lado do leitor *RFID* a indicar que a sessão já iniciou. Caso contrário, um led vermelho piscará, também ao lado do leitor *RFID*, de modo a indicar que o cartão não está atribuído a nenhum paciente **(G)**. Após o início da sessão, a plataforma de computação montada no *smart walker* calcula as métricas constantemente (ex: a elevação do andarilho), e envia-as para o servidor através de um módulo *Wi-Fi* que realiza pedidos *HTTP* através do *script PHP* adequado, de maneira a inserir as métricas da sessão na base de dados do sistema **(H)**. A sessão termina quando o cartão *RFID* volta a ser encostado ao leitor de cartões *RFID*.

No segundo caso, o fisioterapeuta, utilizando a *PhysioApp*, dá início à sessão de fisioterapia. Ao iniciar a sessão de fisioterapia, um pedido *HTTP* é realizado através do ficheiro *PHP* adequado (o *script create_session.php* presente na secção 2.4.3 do Apêndice C), de maneira a registar uma nova sessão de fisioterapia para o paciente em causa, na presente data. Posteriormente é realizada pela *PhysioApp* uma comunicação *Bluetooth* com o andarilho, de modo a que esse dispositivo receba as métricas calculadas pelo aparelho de fisioterapia **(I)**. Os dados recebidos são visualizados em tempo real em gráficos apropriados, podendo ser vistos pelo fisioterapeuta **(J)**, ao mesmo tempo que são inseridos na base de dados do sistema, por intermédio de pedidos *HTTP* e acedendo ao ficheiro *PHP* criado para o efeito **(K)**. O fisioterapeuta fica com a responsabilidade de terminar a sessão ao clicar no botão "*stop session*" presente na aplicação. Com o término da sessão é realizado um novo pedido *HTTP* através do ficheiro *PHP* correspondente de maneira a atualizar a hora do fim da sessão.

Finalmente, e após a sessão de fisioterapia ser concluída, o fisioterapeuta poderá aceder aos dados da sessão de fisioterapia realizada, avaliá-los e ajustar o plano de exercícios, se necessário. Quando o paciente tiver realizado mais de uma sessão, o fisioterapeuta poderá ver a comparação destas sessões por data, com estatísticas para avaliar a evolução do paciente durante o plano de tratamento **(L)**.

3.4. Componentes de *hardware*

Os componentes de hardware utilizados para o desenvolvimento do sistema *IoPhyR* dividem-se em 4 categorias:

- Andarilho.

- Módulo *PhysioRegister*.
- Servidor.
- *Tablet* com o sistema operativo *Android*.

Andarilho

O andarilho é o nó da rede desenvolvida para o sistema *IoPhyR*. A sua utilização é destinada aos pacientes sob a supervisão dos fisioterapeutas. Em relação à sua composição, este é composto por um microcontrolador que calcula métricas de orientação e equilíbrio do paciente, número de passos dados, elevação bilateral e força exercida do nos pés do andarilho que são disponibilizadas em tempo real ao fisioterapeuta e armazenadas no servidor do sistema.

Este elemento é abordado mais detalhadamente no Capítulo 4.

Módulo *PhysioRegister*

O *IoPhyR* conta com um módulo que auxilia os fisioterapeutas no registo de pacientes. Para isso, o *PhysioRegister*, realiza a leitura e envio dos identificadores de cartões *RFID* para a *PhysioApp* aquando da criação de um novo paciente.

Este módulo, apresentado na Figura 3.5, inclui uma plataforma de desenvolvimento *Arduino Nano, uma interface de comunicação Bluetooth*, e um leitor *RFID* para ler os identificadores de cartões associados aos utilizadores do andarilho inteligente. Após a leitura, o *PhysioRegister* espera que a *PhysioApp* realize uma comunicação com o seu módulo *Bluetooth* para enviar o identificador do cartão lido para o formulário de criação de pacientes.

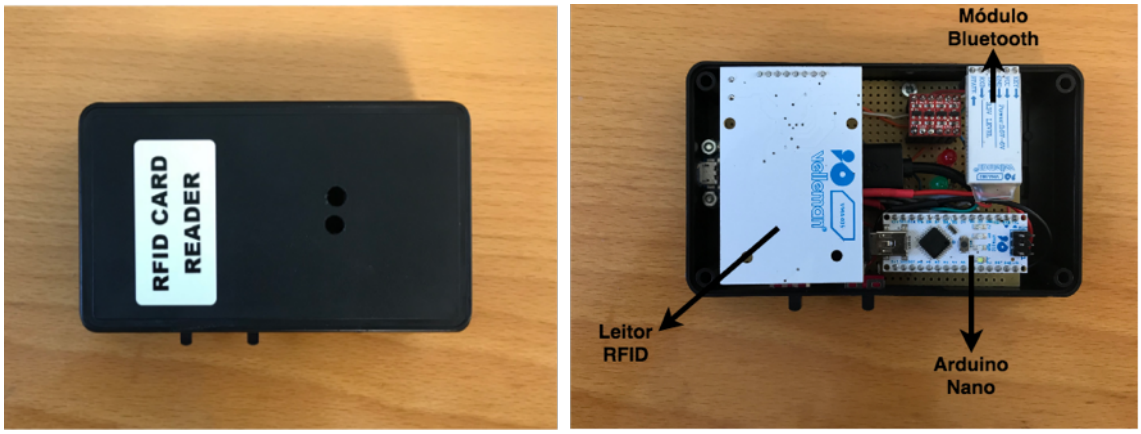


Figura 3.5 – Módulo *PhysioRegister*.

Servidor

O sistema desenvolvido e descrito na presente dissertação conta com o *PhysioWatch*, um servidor alojado na *Cloud*. O acesso ao mesmo é feito por *SSH* (*Secure Shell*)

utilizando o seu *IP* público e as suas credenciais (*username* e *password*). A configuração do mesmo é a *LAMP*, sendo o *Linux* o sistema operativo, com uma distribuição *Debian*, o *Apache* o *webserver* para realizar comunicações *HTTP*, o *MySQL* o gestor de base de dados e o *PHP* a linguagem de *scripting* para comunicar com a base de dados do sistema. No servidor estão alojados a base de dados, os ficheiros *PHP* que comunicam com ela, e a *PhysioWebapp,* a aplicação *web* do sistema. O servidor está amplamente descrito no Capítulo 5.

Tablet Android

A aplicação móvel do sistema, a *PhysioApp*, foi desenvolvida para utilizadores que possuam um dispositivo com o sistema operativo *Android*, podendo este ser um *smartphone* ou um *tablet*. Entre as funcionalidades da aplicação móvel são indicadas: a visualização de dados das sessões dos pacientes, a criação de planos de exercícios e e a visualização de dados de sessões em tempo real.

Esta aplicação foi desenvolvida, utilizando Java sobre a ferramenta Android Studio, para dispositivos com o sistema operativo *Android* com a versão 5.0 (*Lollipop*). que permite que corra em dispositivos desde essa versão até aos mais recentes, podendo por isso ser instalada por 71,3% dos dispositivos activos no mundo inteiro, como se pode ver na Figura 3.6.

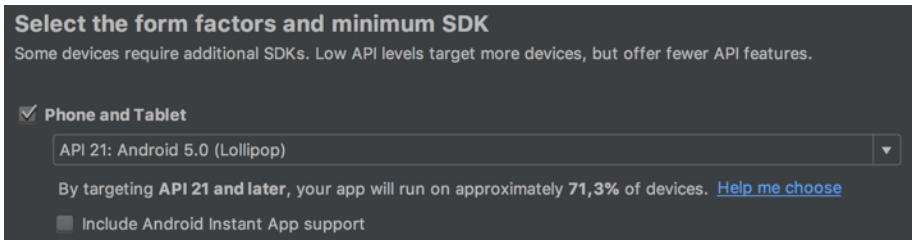


Figura 3.6 – *API* mínima suportada, e percentagem de dispositivos alcançados.

Foram realizados ensaios da aplicação em diferentes tipos de dispositivos incluindo um *tablet* (*Samsung Tab S2*) e um *smartphone* (*Google Nexus 5*).

- O *Samsung Tab S2* é um *tablet* de 9.7'', com uma resolução *QXGA* (2048x1536 pixeis), com um processador *Exynos 5433*, 3 GB de memória *RAM*, 32 GB de memória *ROM*, com *Wi-Fi* e *LTE*, e com a versão 7.0 do *Android* (*Nougat*).
- O *Google Nexus 5* é um smartphone de 4.95'', com uma resolução de 1080x1920 pixeis, com um processador *Qualcomm Snapdragon 800*, 2 GB de memória *RAM*, 16 GB de memória *ROM*, com *Wi-Fi* e *LTE*, e com a versão 5.0 do *Android* (*Lollipop*).

Capítulo 4 – Andarilho

Neste capítulo pretende-se apresentar o prótotipo de andarilho desenvolvido, o *hardware*, o *software* embebido e o seu funcionamento.

4.1. O andarilho e o seu funcionamento

O protótipo de andarilho inteligente, apresentado na Figura 4.1, representa o componente principal do sistema *IoPhyR*. Este é utilizado pelos pacientes sob a supervisão dos seus fisioterapeutas para realizar os treinos da reabilitação da marcha. Treinos esses que são iniciados através da leitura do cartão do paciente, ou diretamente na aplicação móvel do sistema, a *PhysioApp*. O andarilho inteligente inclui uma plataforma com microcontrolador que calcula as métricas de orientação e equilíbrio do paciente, de elevação bilateral do andarilho, do número de passos dados, e das forças exercidas sobre os pés do andarilho. Caso a sessão seja iniciada a partir da PhysioApp, o fisioterapeuta pode visualizar os dados calculados em tempo real, através do protocolo *Bluetooth*. Por outro lado, se esta for iniciada com o cartão do paciente, os dados só poderão ser consultados após o fim da mesma.

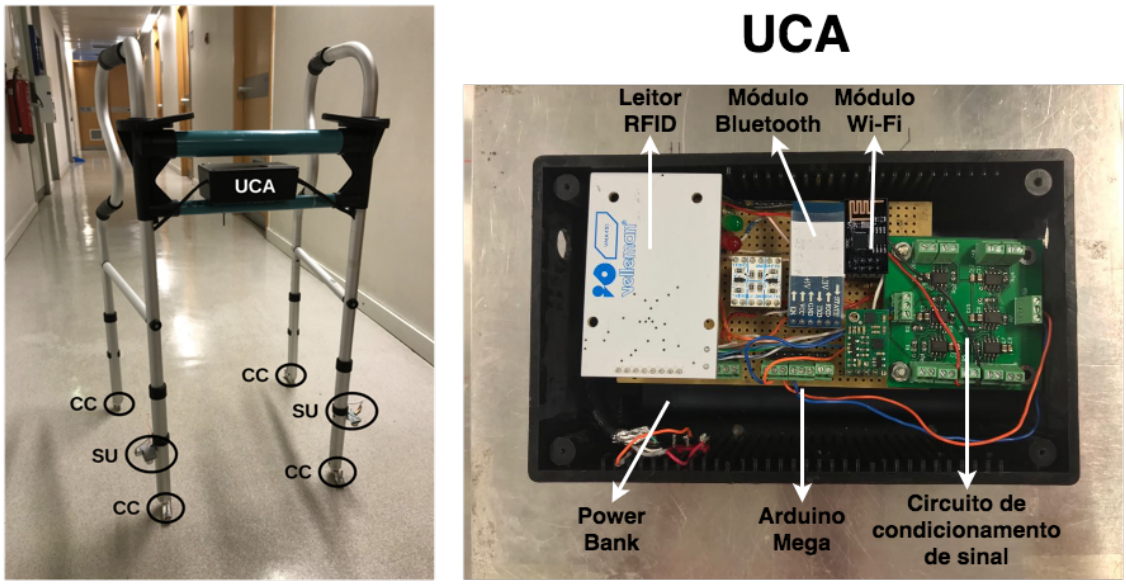


Figura 4.1 – Protótipo de andarilho desenvolvido (esquerda);*UCA* do andarilho (direita).

4.2. *Hardware* embebido

A Figura 4.2 apresenta o diagrama da arquitetura e componentes de *hardware* do protótipo desenvolvido. O protótipo inclui uma plataforma de desenvolvimento, de tipo *Arduino Mega*, oito sensores de três tipos diferentes, um circuito de condicionamento de sinal, um módulo de comunicação *Bluetooth* e outro de comunicação *Wi-Fi*. São usados

um sensor de orientação (*IMU*), um leitor *RFID*, dois sensores de distância (sensores ultrassónicos), e quatro sensores de força (células de carga). Em relação à sua localização no andarilho, o sensor de orientação, o leitor *RFID* e os módulos de *Wi-Fi* e *Bluetooth* e o circuito de condicionamento, estão colocados juntamente com a plataforma de desenvolvimento, na unidade central do andarilho, apresentada na Figura 4.1 com o termo "*UCA*". Os sensores de distância são colocados nos dois pés frontais do andarilho (a 5 cm do chão), assinalados na mesma figura com a letra "S*U*". Finalmente, os sensores de pressão estão colocados por baixo dos 4 pés do andarilho, assinalados nesta figura com as letras "*CC*".

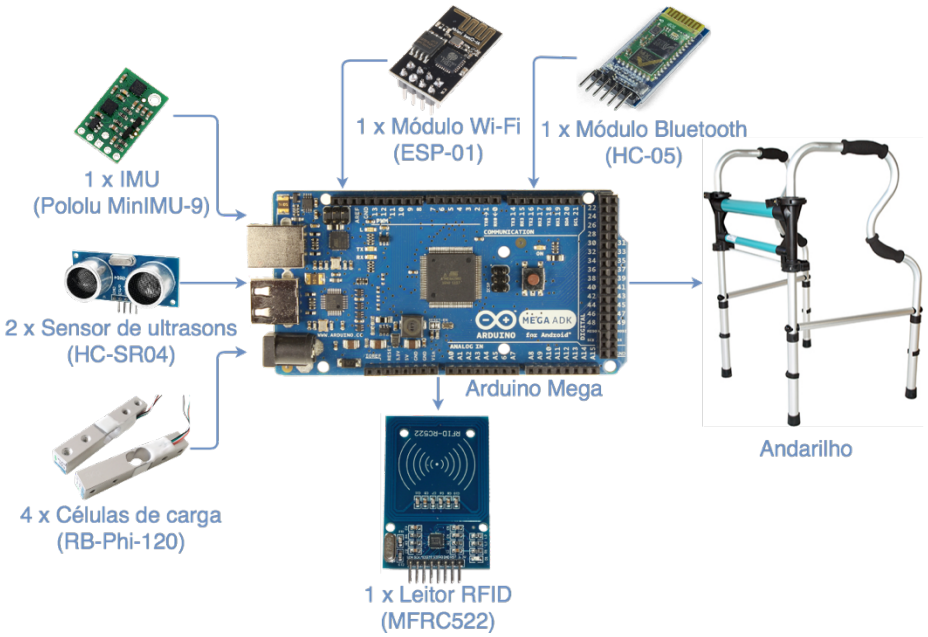


Figura 4.2 – Diagrama da arquitetura do nó do andarilho.

4.2.1. Plataforma de desenvolvimento

O *Arduino Mega*, apresentado na Figura 4.3, foi a plataforma de desenvolvimento escolhida para integrar o andarilho. Essa escolha deveu-se ao facto desta ser uma solução potente e barata, com portas digitais e analógicas que contém diversas bibliotecas criadas para as diferentes utilidades, e que tem uma comunidade de utilizadores muito activa na internet, que reportam erros e soluções para os mesmos.

Em relação às suas características, o *Arduino Mega*, contém um microcontrolador *ATMega 2560*, 54 pinos digitais de entrada ou saída (podendo 15 deles ser usados como saídas *PWM)*, 16 entradas analógicas e quatro *UART's* (portas *serial* de *hardware*). Em relação às memórias, uma *Flash* de 256 KB (sendo usados 8 pelo *bootloader*), uma *SRAM* de 8 KB, com uma *EEPROM* de 4 KB, com um cristal oscilador de 16 MHz. A comunicação pode ser realizada usando o protocolo *I2C*, *UART* e *SPI* (através do

cabeçalho *ICSP* presente na plataforma). Finalmente, A alimentação pode ser realizada através com uma porta *USB* (que também serve para carregar o código), uma entrada de energia que pode receber entre 6 e 20 V, ou ligando uma bateria aos pinos terra (*Ground*) e *Vin* [52]. Sendo que no caso específico do protótipo desenvolvido foi usada uma *powerbank* de 10000 mAh, com uma tensão de output de 5V, e uma corrente de 2A, o que garante, segundo testes realizados, um dia de utilização.

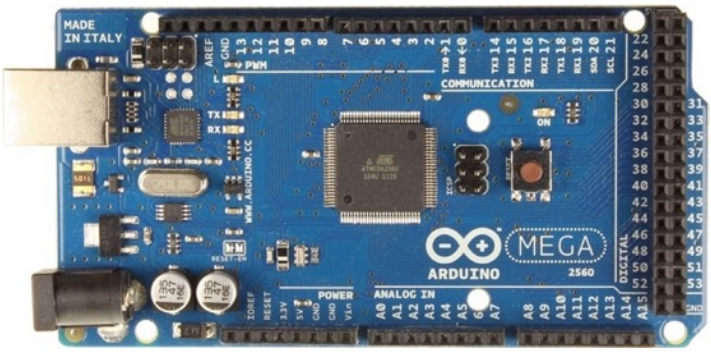


Figura 4.3 – Plataforma de desenvolvimento *Arduino Mega*.

4.2.2. Sensor *IMU*

A unidade de medição inercial (sensor *IMU)* utilizada, foi o *Pololu MinIMU-9 v3,* apresentado na Figura 4.4, que é uma pequena e compacta placa que conjuga um giroscópio (*L3GD20H*), e um acelerómetro com magnetómetro (*LSM303D*) cada um com 3 eixos, atribuindo a este sensor 9 graus de liberdade no total. Este sensor recebe 3.3V de entrada, opera a uma voltagem no intervalo [2.5;5.5] V e uma corrente de 6 mA, utiliza ainda o *SPI* ou o *I2C* como interfaces para comunicar com a placa microcontroladora à qual está ligada [53]. As leituras do *IMU* são realizadas utilizando a interface de comunicação *I2C*. Estas tem um tamanho de 16 *bits* por cada eixo do acelerómetro, magnetómetro e giroscópio [53]. Este sensor é utilizado no protótipo desenvolvido para calcular a orientação do andarilho durante os treinos de marcha com pacientes já identificados utilizando a tecnologia RFID.

Os níveis de sensibilidade do giroscópio acelerómetro e magnetómetro são os seguintes:

- Giroscópio: [-245;245] *%s*, [-500;500] *%s* ou [-2000; 2000] *%s*.
- Acelerómetro: [-2;2] *g*, [-4;4] *g*, [-6;6] *g*, [-8;8] *g*, ou [-16;16] *g*.
- Magnetómetro: [-2;2] *gauss*, [-4;4] gauss, [-8;8] *gauss* ou [-12;12] *gauss*.

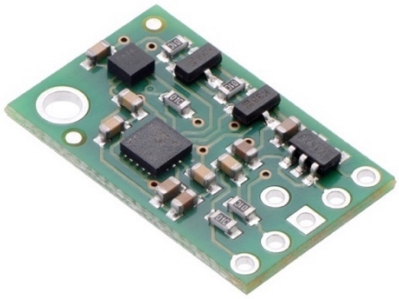


Figura 4.4 – Sensor *IMU* utilizado (*Pololu MinIMU-9 v3*).

4.2.3. Sensor de ultrassons

Os dois sensores de ultrassons utilizados, com referência *HC-SR04*, são apresentados na Figura 4.5. Eles permitem determinar as distâncias entre os próprios sensores e os obstáculos sempre e quando a estas esteja compreendida no intervalo [2; 400] cm. Estes sensores operam com uma tensão de 5V, uma corrente de 15 mA e uma frequência de 40 KHz [54], [55].

Figura 4.5 – Sensor de ultrassons *HC-SR04*.

Este tipo de sensor opera emitindo 8 pulsos de 40KHz após ser enviado um sinal com duração de 10 microsegundos para o seu pino *Trigger*, e esperando depois receber o som refletido pelo obstáculo no receptor, medindo o tempo no qual o pino echo do sensor permanece no estado *HIGH* para calcular a distância à qual este se encontra do referido obstáculo, como exemplificado na Figura 4.6. Esta medição é possível sempre e quando a onda ultrassónica emitida tenha um ângulo de reflexão inferior a 15 graus [54].

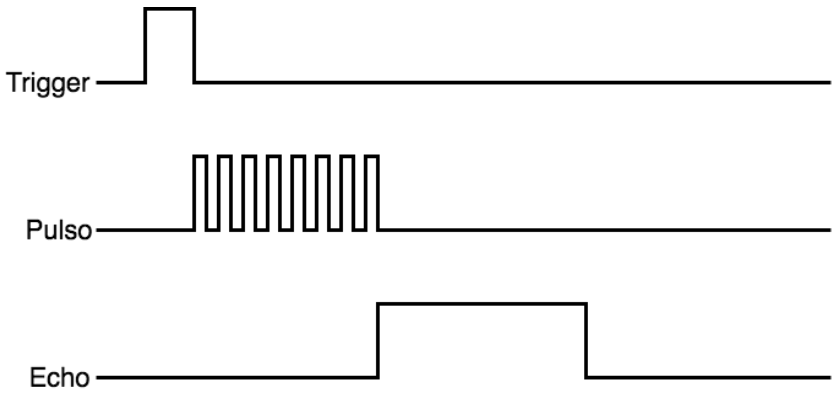


Figura 4.6 – Diagrama de funcionamento do sensor *HC-SR04*.

4.2.4. Células de carga

Os quatro sensores de força utilizados, com referência *RB-Phi-120*, apresentados pelo exemplar da Figura 4.7, são pequenos sensores conhecidos como células de carga, ou *load cells*. Estes sensores têm extensómetros ou *strain gauges* em zonas específicas da sua estrutura metálica de modo a sentir e mensurar as deformações mecânicas da sua estrutura e consequentemente medir a força exercida num único sentido, ignorando os outros [56].

Figura 4.7 – Célula de carga (*RB-Phi-120*).

Precisão	0.05%
Output	1.0+- 0.15mv/V
Não linearidade	0.05% da Full Scale
Histérise	0.05% da Full Scale
Temperatura de funcionamento	[-20;55] ºC
Capacidade	500 N

Tabela 1 - Características da célula de carga.

Devido ao baixo nível da tensão na saída de sinal destes sensores, verificada na Tabela 1, torna-se necessário o uso de amplificação, utilizando para isso o circuito de condicionamento de sinal da Figura 4.8.

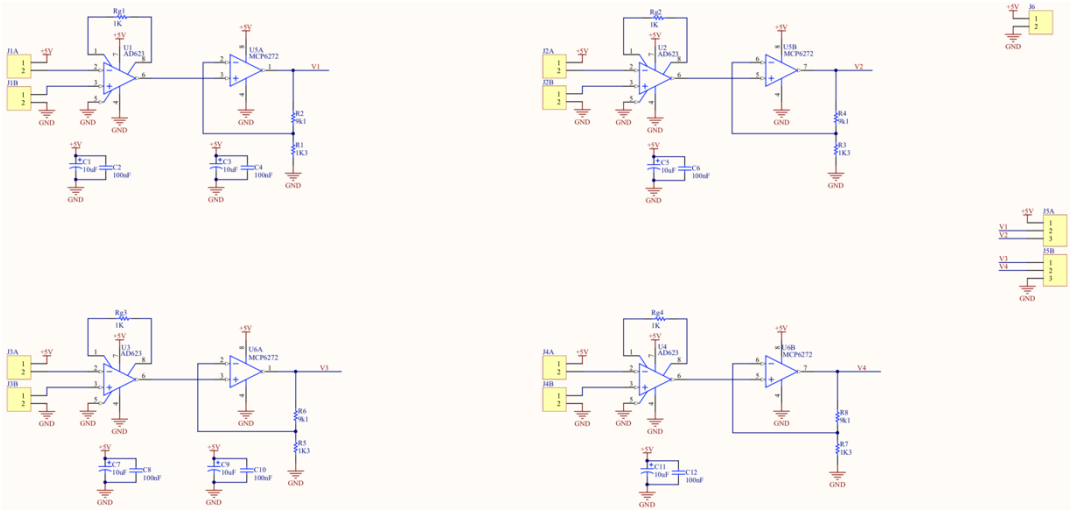


Figura 4.8 – Esquema do circuito de condicionamento de sinal.

Este circuito, o da Figura 4.8, tem como finalidade amplificar o sinal de saída de quatro sensores de células de carga iguais. A amplificação dos sensores é representada, neste circuito, por cada um dos andares de amplificação apresentados. Cada andar de amplificação utiliza um amplificador de instrumentação (*AD623*), assim como um amplificador não inversor (*MCP6272*). O primeiro tem um ganho de 100 e o segundo de 8, perfazendo um ganho total de 800 por andar.

Este circuito é alimentado com 5V e para cargas entre 0 e 500 N, as tensões *Vaux* variam entre 0 e 500 mV e as tensões de saída variam entre 0 e 4V.

De modo a conseguir estabelecer-se uma corelação entre a tensão de saída de cada sensor com a força em *N* aplicada no mesmo, foram realizadas sessões de calibração para encontrar as equações características dos quatro sensores de força. As sessões consistiram em realizar força sobre cada um dos sensores quando este estava por cima de uma balança. Foram realizadas medições da tensão de saída dos sensores para 5, 10, 150 e 200 N obtendo-se equações características com um baixo desvio das amostras, como se pode ver na Figura 4.9.

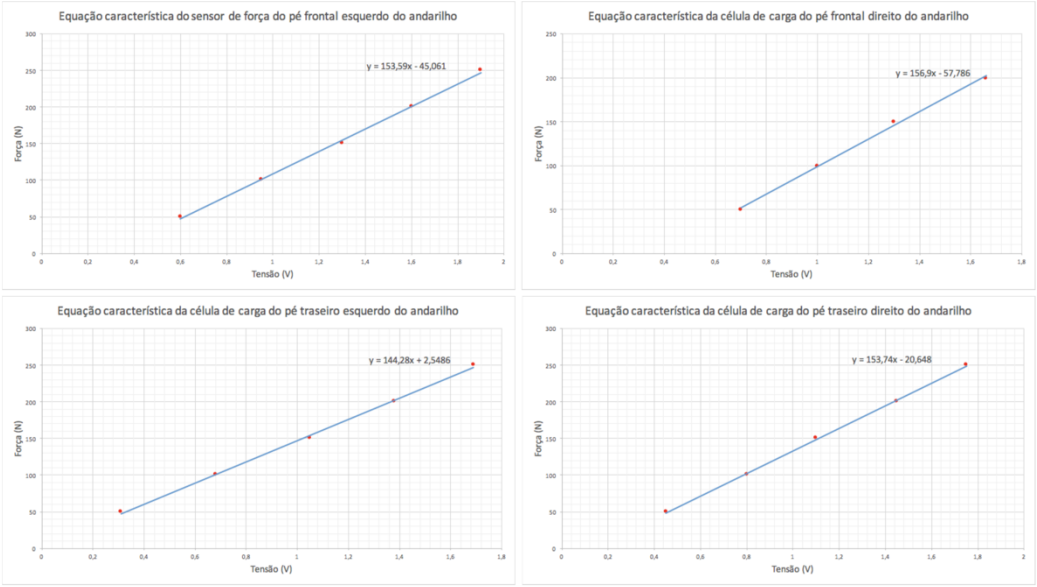


Figura 4.9 – Equações características dos sensores de força. Pé frontal esquerdo (canto superior esquerdo); Pé frontal direito (canto superior direito); Pé transeiro esquerdo (canto inferior esquerdo); Pé transeiro direito (canto inferior direito).

4.2.5. Leitor *RFID*

O leitor *RFID* utilizado, o *MFRC522* que é apresentado na Figura 4.10, lê e escreve em *RFID* passivos como cartões e *tags* a 13.56 MHz e até 50 mm de distância. O leitor conta com um módulo de receção robusto e eficiente que trata da desmodulação e descodificação dos sinais recebidos dos *RFID* passivos tendo mecanismos de correção de erros. Estes *RFID* passivos contêm, cada um, 64 *bytes* de informação onde está incluído o seu *ID* e outros campos onde se pode escrever para adicionar informação.

Em relação à comunicação do leitor RFID com um microcontrolador ou computador, estão disponíveis as interfaces *Serial Peripherical Interface* (*SPI*), *Universal Asynchronous Receiver/Transmitter(UART*) ou o *I2C* para a comunicação com o *Master* [57]. Em relação à alimentação, este sensor utiliza uma tensão de 3.3 V e uma corrente no intervalo de [9,10] µA em funcionamento normal e 100 mA em transmissão [57].

Este sensor apresenta-se como solução simples e barata para a identificação de pacientes neste sistema, fazendo com que os pacientes só se tenham de fazer acompanhar de um cartão ou tag *RFID* para realizarem as sessões.

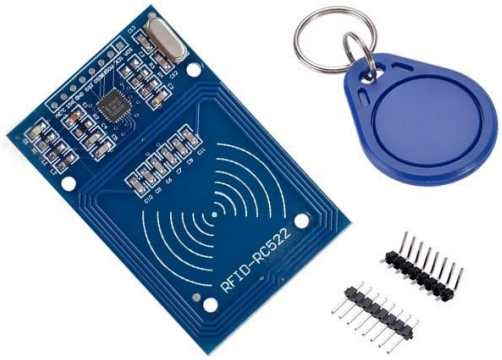


Figura 4.10 – Leitor de cartões *RFID*.

4.2.6. Módulo *Bluetooth*

Este sensor, o *HC-05*, que é apresentado na Figura 4.11, é um módulo de *Bluetooth* pequeno, de baixo consumo e de fácil implementação que utiliza o protocolo *SPP* (*Serial Port Protocol*) para comunicar. O módulo permite ter comunicações *Full Duplex*, podendo enviar e receber dados e mudar entre os modos *master* e *slave*. Este módulo utiliza a especificação *Bluetooth* V2.0 + *EDR* (*Enhanced Data Rate*), o que indica que é indicado para ser usado em curtas distâncias e que conta com 3 Mbps de transmissão, e utiliza um *chip CSR Bluecore4* com tecnologia *CMOS* e com *AFH* (*Adaptive Frequency Hopping*) [58].

Em relação ao consumo tem uma potencia de funcionamento de 1.8 V, sendo aceita um input no intervalo de [3.6;6] V. A sua sensibilidade é de -80 dBm, tem uma potência de transmissão de até +4 dBm, tem uma interface *UART* com possibilidade de programar diferentes *baud rates* e tem uma antena integrada [58].

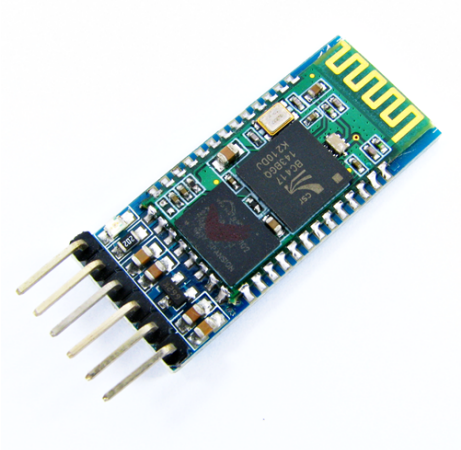


Figura 4.11 – Módulo *Bluetooth HC-05*.

4.2.7. Módulo de *Wi-Fi*

Este sensor, o *ESP-01* que é apresentado na Figura 4.12, é um módulo *Wi-Fi* pequeno, popular e barato, que confere a qualquer microcontrolador ao qual esteja ligado acesso a redes *Wi-Fi*. Este módulo conta com o protocolo *TCP/IP* integrado, suporta as normas 802.11 b, g e n, assim como o *Wi-Fi-Direct P2P*, standard do *Wi-Fi*. Pemite ainda o uso de mecanismos ou métodos como o *STBC*, e *MIMO* 1x1 e 2x1, de forma a minimizar erros e optimizar a velocidade de transferência. O seu processador de 32 *bits* e de baixa potência (< 1.0mW em stand-by), e a sua memória *Flash* de 1 MB permitem que seja usado em conjuto com sensores e outras aplicações utilizado os *GPIO* (*General Porpose Input Output*). Este módulo suporta ainda o mecanismo *APSD* (*Automatic Power Save Delivery*) para aplicações *VoIP*, interfaces para a coexistência com o *Bluetooth*, e as interfaces *SDIO* 1.1 e 2.0, *SPI* e *UART* para a comunicação com os dispositivos aos quais se pretende ligar, como microcontroladores. Finalmente, em relação à potência de entrada necessária, o *ESP-01*, apesar de poder ser ligado com 3.3V, é requerido um divisor resitivo ou um conversor de nível lógico, uma vez que este não é capaz de oscilar sozinho entre 5V e 3.3V a potência recebida, conforme as necessidades [59].

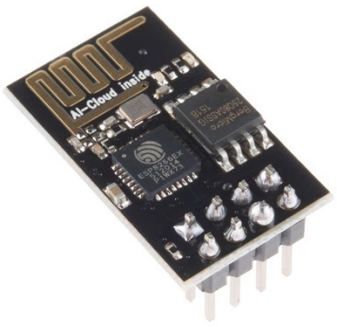


Figura 4.12 – Módulo *Wi-Fi ESP-01*.

4.3. *Software* embebido

O *software* embebido utilizado na plataforma *Arduino* foi desenvolvido em *C* e no *Arduino IDE*, um ambiente de desenvolvimento *open source* disponibilizado pela empresa com o mesmo nome (ver Figura 4.13). Esta ferramenta permite programar o *Arduino* utilizando as duas funções principais, a *setup()* e a *loop()*. A primeira é processada uma única vez, logo no início da execução do programa. A segunda é executada continuamente após a conclusão da primeira função.

Figura 4.13 – Captura do *Arduino IDE*.

As métricas calculadas neste *software* são a orientação do andarilho, a elevação bilateral do andarilho, o número de passos realizados, a força exercida sobre cada pé do andarilho e o equilíbrio do paciente (centro de forças). Estas métricas podem ser consultadas nas subseções seguintes (4.3.1 a 4.3.5).

4.3.1. Cálculo da orientação do andarilho

A orientação do andarilho, utilizado pelo paciente durante os treinos de marcha, é calculada a partir dos valores extraídos do acelerómetro, magnetómetro e giroscópio presentes no sensor do tipo *IMU*, utilizando o protocolo de comunicação *I2C*. Estes valores são obtidos através da utilização das bibliotecas *L3G* [60] e *LSM303* [61], que interagem com o giroscópio e magnetómetro e acelerómetro do sensor, respetivamente.
Com os dados extraídos do sensor pode obter-se os ângulos de orientação do andarilho, ou ângulos de *Euler*. Estes ângulos, apresentados na Figura 4.14, são os ângulos de rotação do andarilho no eixo dos X (*Pitch*), dos Y (*Roll*) e dos Z (*Yaw*) [62], [63].

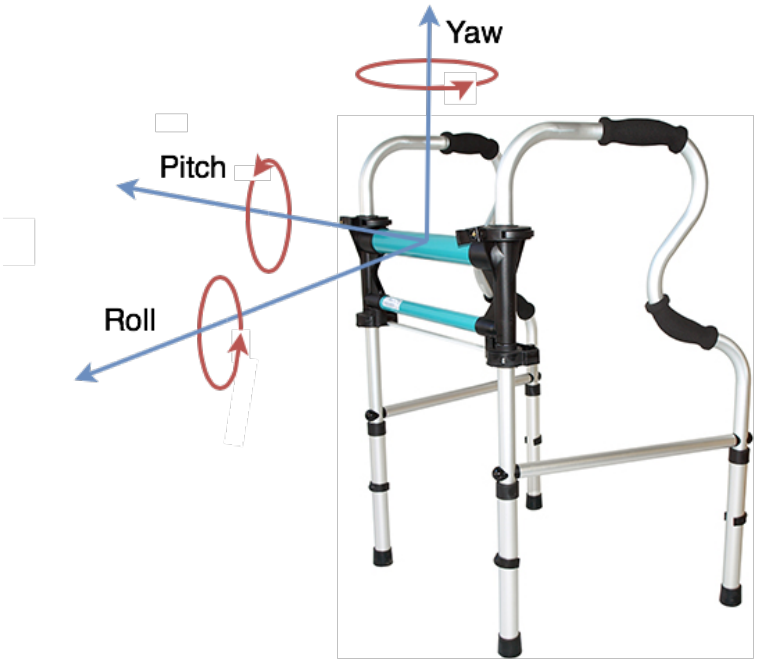


Figura 4.14 – Diagrama de ângulos de *Euler* no andarilho.

De modo a obter os ângulos de *Euler* (*roll*, *pitch* e *yaw*), são usadas as equações (1), (2) e (3) [64], [65]. Estas equações relacionam os dados adquiridos do acelerómetro, magnetómetro e giroscópio presentes no sensor *IMU* para obter os ângulos desejados.

A equação (1) é usada para calcular o ângulo *Pitch*. Nesta equação o *P* representa o ângulo *Pitch*, o a_x é a aceleração no eixo dos *X*, e *g* é a constante gravitacional (g = *9,80665 m/s*^{2}).

$$P = \operatorname{asin}\left(\frac{-a_x}{g}\right) \tag{1}$$

A equação (2) é usada para calcular o ângulo *Roll*. Nesta equação o *R* representa o ângulo *Roll*, a_y é a aceleração no eixo dos Y, e a_z é a aceleração no eixo dos *Z*.

$$R = \operatorname{atan2}\left(\frac{a_y}{a_z}\right) \tag{2}$$

A equação (3) é utilizada para obter o ângulo *Yaw*. Nesta 3ª equação o *Y* representa o ângulo *Yaw*, o m_x e m_y os valores de campo magnético no eixo dos *X* e dos *Y*, respetivamente.

$$Y = \operatorname{atan2}\left(\frac{m_y}{m_X}\right) \tag{3}$$

De modo a poder-se obter valores mais estabilizados dos ângulos de *Euler*, implementou-se um filtro de *Kalman*, através da utilização de uma biblioteca para o *Arduino* em *C*, a *KalmanFilter* [66].

Estas equações são utilizadas na função *getEulerAngles()* apresentada na Figura 4.15, e desenvolvida em *C* no *Arduino IDE*. Na referida função, são realizadas primeiramente as leituras do giroscópio (linha 403) e do magnetómetro e acelerómetro (linha 405). De seguida é calculado o tempo actual e a orientação do eixo dos *ZZ*, tendo em conta o valor de aceleração no mesmo eixo (linhas 407 a 414). Posteriormente são calculados os valores do acelerómetro, magnetómetro e giroscópio nos três eixos usando os valores extraídos e as características de cada sensor. Sendo que a unidade destes valores é *g* para o acelerómetro, *gauss* para o magnetómetro e *dps* para o giroscópio (linhas 416 a 438). A seguir, são calculados os ângulos de *Euler* (*roll*, *pitch* e *yaw*) utilizando funções trigonométricas e os valores de acelerómetro e magnetómetro (linhas 441 a 444). Finalmente, os ângulos de orientação calculados são suavizados com um filtro de *Kalman* utilizando a biblioteca *KalmanFilter*, os ângulos de orientação calculados e os valores obtidos do giroscópio nos três eixos (linhas 446 a 449).

```
void getEulerAngles(unsigned long currentMillis, unsigned long previousMillis){
    /*Read gyro values*/
    gyro.read();
    /*Read accelerometer and magnetometer values*/
    compass.read();
    /*get current time in seconds*/
    timestamp = currentMillis/1000;
    /*Establish Z axis orientation*/
    if (compass.a.z < 0){
            signOfz = -1;
    }
    else{
            signOfz = 1;
    }

    /* Accelerometer characteristics
    *16-bit, default range +-2 g, sensitivity 0.061 mg/digit
    */
    /*3 axis accelerometer calculus*/
    aX = (double)(compass.a.x)*0.061/1000.0;// g
    aY = (double)(compass.a.y)*0.061/1000.0;// g
    aZ = (double)(compass.a.z)*0.061/1000.0;// g

    /* Magnetometer characteristics
     * 16-bit, default range +-2 gauss, sensitivity 0.080 mgauss/digit
    */
    /*3 axis magnetometer calculus*/
    mX = (double)(compass.m.x)*0.080/1000.0;// gauss
    mY = (double)(compass.m.y)*0.080/1000.0;// gauss
    mZ = (double)(compass.m.z)*0.080/1000.0;// gauss

    /*Gyroscope characteristics
    *16-bit, default range +-245 dps (deg/sec), sensitivity 8.75 mdps/digit
    */
    /*3 axis gyroscope calculus*/
    gX = (double)(gyro.g.x)*8.75/1000.0;// dps
    gY = (double)(gyro.g.y)*8.75/1000.0 * signOfz;// dps
    gZ = (double)(gyro.g.z)*8.75/1000.0;// dps


    /*Orientation angles calculus*/
    roll  = asin(-aX, g)*180/M_PI;
    pitch = atan2(aY, aZ)*180/M_PI;
    yaw   = atan2(mY,mX);

    /*Kalman Filter applied to the orientation angles*/
    roll  = kalmanX.getAngle(roll, gX, currentMillis - previousMillis);
    pitch = kalmanY.getAngle(pitch, gY, currentMillis-previousMillis);
    yaw   = kalmanZ.getAngle(yaw, gZ, currentMillis-previousMillis);
}
```

Figura 4.15 – Função do cálculo dos ângulos de orientação.

4.3.2. Cálculo da elevação bilateral do andarilho

Neste sistema são usados 2 sensores de ultrasom, colocados um em cada uma das pernas frontais do andarilho de modo a poder obter as elevações laterais do mesmo e consecutivamente detetar assimetrias nas mesmas, assim como o número de passos realizados.

Para o cálculo da elevação é calculada inicialmente a distância dos sensor ao solo, usando a equação (4) de maneira a que a partir desse momento se possa calcular a distância a que o andarilho está do solo, retirando esse valor inicial como mostrado na equação (5) [54], [67].

$$d_{ss} = \left(\frac{t_{ss} * v_s}{2}\right) \tag{4}$$

$$d_{as} = \left(\frac{t_{as} * v_s}{2}\right) - d_{ss} \quad (5)$$

Na equação (4), d_{ss} representa a distância do sensor ao solo, t_{ss} o tempo entre a emissão e receção da onda ultrassónica quando o andarilho está no chão e v_s a velocidade do som (340 m/s). Por outro lado, a equação (5) representa a elevação do andarilho, t_{as} o tempo decorrido entre a emissão e recepção da onda ultrassónica, v_s a velocidade do som (340 m/s) e d_{ss} a distância entre o sensor e o chão calculada anteriormente.

As divisões presentes nas equações (4) e (5) prendem-se com o facto da onda percorrer duas vezes a distância entre o sensor e o obstáculo, como se pode ver na Figura 4.16.

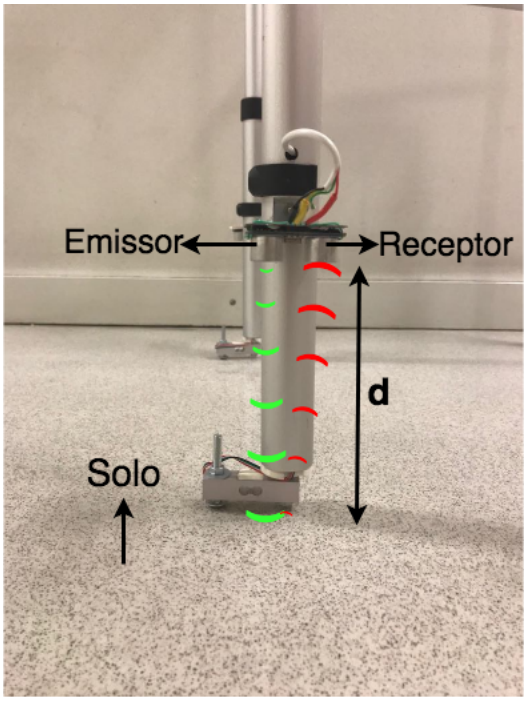


Figura 4.16 – Persurso da onda ultrassónica transmitida pelo sensor *HC-SR04*.

Estas equações são utilizadas na função *getElevations()*, apresentada na Figura 4.17, e desenvolvida em *C* no *Arduino IDE*. Na referida função, é realizado primeiramente *reset* do pino *trigger* do sensor, utilizando a função *digitalwrite()* para lhe fornecer uma tensão de 0V (linha 226). Posteriormente são aguardados 2 microsegundos utilizando a função *delayMicroseconds()* (linha 228). De seguida são usadas as funções *digitalWrite()* e *delayMicroseconds()* para deixar o pino *trigger* do sensor a *HIGH*, isto é com 5V, durante 10 microsegundos (linhas 230 e 232). Após a passagem deste período, é novamente deixado o pino *trigger* do sensor a *LOW*, isto é com 0V (linha

234). De seguida, é calculado o tempo de transmissão da onda ultrassónica em microssegundos, utilizando a função *pulseIn()* com valor *HIGH* (5V), no pino echo do sensor (linha 236). Finalmente, e tendo em conta o referido tempo de transmissão, a distância do sensor ao chão e a velocidade de transmissão de uma onda ultrassónica, é calculada a elevação do andarilho (linha 238).

Nota: Os cálculos realizados nos intervalos de linhas [226-238] e [241-253] são semelhantes, simplesmente são respectivos a cada um dos sensores.

```
223 float getElevations(){
224   /*---Right side---*/
225     /*Clears the trigPin*/
226     digitalWrite(rightTrigPin, LOW);
227     /*Delay 2 microseconds*/
228     delayMicroseconds(2);
229     /*Sets the trigPin on HIGH state for 10 micro seconds*/
230     digitalWrite(rightTrigPin, HIGH);
231     /*Delay 10 microseconds*/
232     delayMicroseconds(10);
233     /*Clears the trigPin*/
234     digitalWrite(rightTrigPin, LOW);
235     /*Reads the echoPin, returns the sound wave travel time in microseconds*/
236     rightDuration = pulseIn(rightEchoPin, HIGH);
237     /*Calculate the distance*/
238     rightElevation= (rightDuration*0.034/2)-avgElevationRight;
239   /*---Left side---*/
240     /*Clears the trigPin*/
241     digitalWrite(leftTrigPin, LOW);
242     /*Delay 2 microseconds*/
243     delayMicroseconds(2);
244     /*Sets the trigPin on HIGH state for 10 micro seconds*/
245     digitalWrite(leftTrigPin, HIGH);
246     /*Delay 10 microseconds*/
247     delayMicroseconds(10);
248     /*Clears the trigPin*/
249     digitalWrite(leftTrigPin, LOW);
250     /*Reads the echoPin, returns the sound wave travel time in microseconds*/
251     leftDuration = pulseIn(leftEchoPin, HIGH);
252     /*Calculate the distance*/
253     leftElevation= (leftDuration*0.034/2)-avgElevationLeft;
254 }
```

Figura 4.17 – Função do cálculo da elevação bilateral do andarilho.

4.3.3. Cálculo do número de passos

Para o cálculo dos passos realizados pelo paciente, são tidas em conta as elevações calculadas pela função *getElevations()* apresentada na subsecção 4.3.2. Para isso foi criada a função *getSteps()*, apresentada na Figura 4.18, que tem em conta a transição de elevação de um valor positivo para zero para incrementar o número de passos.

Nessa função, é verificado primeiramente se os valores de elevação são superiores a 3 cm e se a variável booleana *onStep* é falsa (linha 298). Em caso afirmativo, está a ser dado um passo e por isso a variável *onStep* é verdadeira (linha 300). Caso contrário, isto é, se a variável *onStep* for verdadeira e se as elevações esquerda e direita do andarilho forem iguais a zero, é porque um novo passo foi completado (linha 305). Sendo assim é incrementada a variável que regista o número de passos, a variável *steps*, e é alterado o estado da variável *onStep* para *false* (linhas 307 e 309).

```
296 void getSteps(){
297     /*if there's no steps undergoing and elevation is greater that 3*/
298     if(onStep == false && leftElevation>=3 && rightElevation>=3){
299       /*then a step is undergoing*/
300       onStep = true;
301     }
302     /*if a step is undergoing*/
303     else{
304       /*if a step is undergoing and the elevation is equal to zero*/
305       if(onStep == true && leftElevation==0 && rightElevation==0){
306         /*increment a step to the step variable*/
307         steps++;
308         /*the step is completed*/
309         onStep = false;
310       }
311     }
312 }
```

Figura 4.18 – Função do cálculo dos passos dados pelo paciente.

4.3.4. Cálculo das forças nos pés do andarilho

As forças exercidas pelos pacientes nos pés do andarilho durante as sessões de fisioterapia, são obtidas a partir das tensões de saída de cada um dos quatro sensores de células de carga colocados nos pés do protótipo.

Para o cálculo da força foram usadas as equações (6), (7), (8) e (9) que correspondem às curvas de calibração mostradas na Figura 4.9.

$$F_{FLF} = 153{,}59 * x - 45{,}061 \quad (6)$$

$$F_{FRF} = 156{,}9 * x - 57{,}786 \quad (7)$$

$$F_{BRF} = 153{,}74 * x - 20{,}648 \quad (8)$$

$$F_{BLF} = 144{,}28 * x - 2{,}5486 \quad (9)$$

Com as equações (6), (7), (8) e (9) obtém-se a força em *N* exercida em cada pé do andarilho a partir da tensão de saída, *x*, de cada sensor em *Volts*. Nestas equações F_{FLF}, F_{FRF}, F_{BRF} e F_{BLF} representam as forças exercidas no pés frontal esquerdo, frontal direito, traseiro direito e traseiro esquerdo do andarilho, respectivamente.

Estas equações são utilizadas na função *getFeetPressure()* apresentada na Figura 4.19, e desenvolvida em *C* no *Arduino IDE*. Na referida função, são guardados em 4 variáveis os valores analógicos adquiridos dos sensores de força colocados nos pés do andarilho (linhas 352 a 355). Estas variáveis adoptam valores no intervalo [0-1023] que tem uma

correspondência linear as tensões no intervalo [0-4] *V*. Posteriormente realiza-se a conversão destes valores para *Volts*, utilizando a referida relação entre os valores analógicos e a tensão (linhas 357 a 360). Finalmente são usadas as equações características dos sensores, previamente calculadas, para converter os valores de *Volts* de cada sensor de força em *N* (linhas 362-365).

```
350 void getFeetPressure(){
351   /*Get load cells analog raw data*/
352   int sensorValue_FLL = analogRead(A0);
353   int sensorValue_FRL = analogRead(A1);
354   int sensorValue_BRL = analogRead(A2);
355   int sensorValue_BLL = analogRead(A3);
356   /*Convert load cells data to Volts*/
357   float FLL_voltage = sensorValue_FLL * (4.0 / 1023.0);
358   float FRL_voltage = sensorValue_FRL * (4.0 / 1023.0);
359   float BRL_voltage = sensorValue_BRL * (4.0 / 1023.0);
360   float BLL_voltage = sensorValue_BLL * (4.0 / 1023.0);
361   /*Convert load cells data to Newtons*/
362   FLL_pressure = (153.59*FLL_voltage) - 45.061;
363   FRL_pressure = (156.9*FRL_voltage) - 57.786;
364   BRL_pressure = (153.74*BRL_voltage) - 20.648;
365   BLL_pressure = (144.28*BLL_voltage) - 2.5486;
366 }
```

Figura 4.19 – Função do cálculo das forças sobre os pés do andarilho.

4.3.5. Cálculo do equilíbrio do paciente

O equilíbrio de um paciente durante as sessões de fisioterapia é obtido a partir das equações (10) e (11) , tendo em conta as forças exercidas por esse mesmo paciente em cada pé do andarilho [23].

$$PCX = \frac{\sum_{K=1}^{K=4} F_K * X_K}{\sum_{K=1}^{K=4} F_K} \tag{10}$$

$$PCY = \frac{\sum_{K=1}^{K=4} F_K * Y_K}{\sum_{K=1}^{K=4} F_K} \tag{11}$$

As equações (10) e (11) representam respectivamente as coordenadas "*x*" e "*y*" do centro das 4 forças aplicadas em pontos com coodenadas (X_K,Y_K). Estas equações relacionam a força exercida em cada ponto com a sua coordenada *x* ou *y* através da utilização de sumatórios.

Quando o andarilho está aberto, os seus pés formam um trapézio isósceles no chão. Sendo assim as coordenadas de cada pé do andarilho, são as dos vétices do trapézio referido, que têm as coordenadas P_1, P_2, P_3 e P_4 como se pode ver na Figura 4.20.

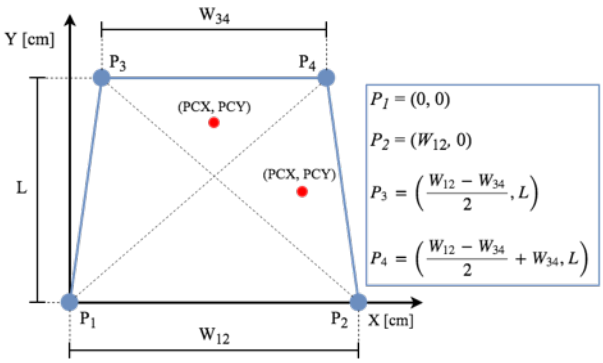


Figura 4.20 – Diagrama da posição geométrica dos pés do andarilho.

Aplicando as coordenadas apresentadas na Figura 4.20 às equações (10) e (11) obtém-se as equações das coordenadas do centro de forças relativo à estrutura do andarilho, onde W_{34} corresponde à distâcia entre os pés 3 e 4, W_{12} à distância entre os pés 1 e 2, e L à diferença das coordenadas "*y*" dos pés 1 e 3:

$$PCX = \frac{(F_3 + F_4) * \frac{W_{12} - W_{34}}{2} + (F_4 * W_{34}) + (F_2 * W_{12})}{F_1 + F_2 + F_3 + F_4}$$

$$PCY = \frac{(F_3 + F_4) * L}{F_1 + F_2 + F_3 + F_4}$$

Estas equações são utilizadas na função *getFeetPressure()* apresentada na Figura 4.21, e desenvolvida em *C* no *Arduino IDE*. Na referida função, é primeiramente verificado se as forças exercidas nos pés do andarilho são todas iguais a zero (linha 371). Em caso afirmativo o ponto de equilíbrio do paciente é definido como centrado (linhas 373 e 374). Caso contrário, são calculadas as coordenadas do equilíbrio do paciente, utilizando as equações (10) e (11) que têm em conta as forças exercidas sobre os pés do andarilho e as suas coordenadas relativas (linhas 377 a 381).

```
378 void getBalance(){
379     /*forces applied to the walker feet are equal to zero*/
380   if(FLL_pressure==0 && FRL_pressure==0 && BRL_pressure==0 && BLL_pressure==0){
381     /*Balance centered*/
382     centroid_x = 28.1;
383     centroid_y = 20.467;
384   }
385   /*forces applied to the walker feet are different from zero*/
386   else{
387     /*Do the balance calculation*/
388     centroid_x = ((FLL_pressure*6.5)+(FRL_pressure*49.7)+(BRL_pressure*56.2))
389     /(FLL_pressure+FRL_pressure+BRL_pressure+BLL_pressure);
390     centroid_y = ((FLL_pressure*42.8)+(FRL_pressure*42.8))
391     /(FLL_pressure+FRL_pressure+BRL_pressure+BLL_pressure);
392   }
393 }
```

Figura 4.21 – Função do cálculo do equilíbrio do paciente.

Capítulo 5 – Servidor

Neste capítulo pretende-se descrever o servidor do sistema, o *PhysioWatch*, assim como o seu funcionamento e as comunicações estabelecidas entre os nós da rede, sejam eles andarilhos, dispositivos com a *PhysioApp* ou que acedam à *PhysioWebapp*, a aplicação *web* do sistema.

5.1. Arquitetura e estrutura do servidor

O sistema desenvolvido na presente dissertação conta com um servidor na nuvem, o *PhysioWatch*, de modo a que todas as sessões de fisioterapia, todos os planos de exercícios e todos os dados dos utilizadores sejam armazenados para que possam ser acedidos e visualizados sempre que se justifique através das aplicações *web*, móvel e embebida do sistema. O servidor conta com uma base de dados *MySQL* para guardar os dados do sistema, uma aplicação *web*, a *PhysioWebapp*, para que os utilizadores possam visualizar esses mesmos dados e um conjunto de ficheiros *PHP* que interagem com a referida base de dados para inserir/recolher dados da mesma.

O servidor tem o *Linux* como sistema operativo e uma arquitetura *LAMP* que está representada na Figura 5.1.

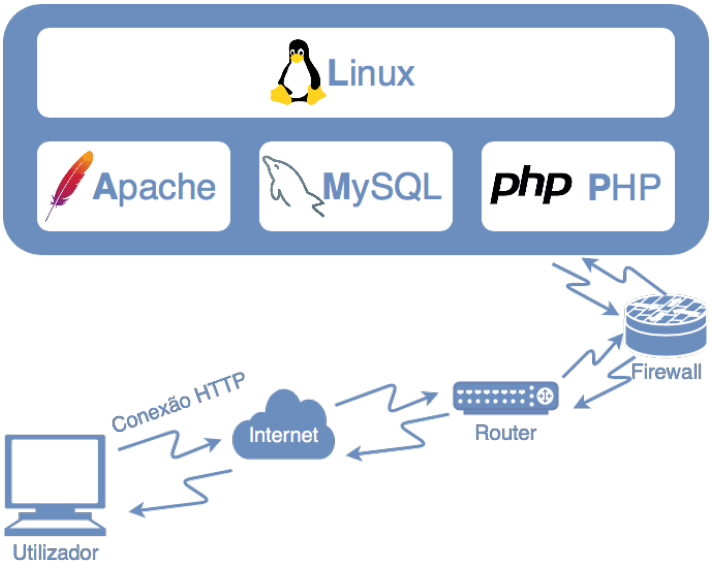


Figura 5.1 – Arquitetura da configuração *LAMP*.

Esta arquitetura é composta por softwares livres, uma vez que está sobre a égide da Fundação *Linux*, o que lhe confere uma sólida confiabilidade e segurança assim como a possibilidade de ser instalada em todos os tipos de *hardware*. A *LAMP* é muito utilizada em servidores *web* e está estruturada em quatro camadas que perfazem a sua sigla, como se pode ver na Tabela 2 [68], [69].

Nome	Descrição
Linux	É o Sistema Operativo da configuração.
Apache	É o Servidor *Web* da configuração.
MySQL	É o sistema de gestão de bases de dados relacionais da configuração.
PHP	É uma linguagem de programação simples e eficiente que permite a interação com bases de dados podendo criar conteúdos dinâmicos.

Tabela 2 – Descrição da configuração *LAMP*.

A primeira letra da sigla da arquitetura do servidor, **L**, corresponde ao sistema operativo que corre no servidor. A segunda, **A**, representa o *Apache*, o servidor *web* utilizado, e permite que a comunicação entre os dispositivos e a base de dados do servidor, escutando e respondendo a pedidos *HTTP*. A terceira, **M**, corresponde ao *MySQL*, o sistema de gestão de bases de dados. E finalmente, a quarta, **P**, representa a linguagem de programação usada para realizar a interação dos clientes com a base de dados do sistema.

Em relação à estrutura organizacional das pastas e ficheiros do sistema no servidor, esta está apresentada na Figura 5.2. Nesta pasta de raíz encontram-se 3 pastas relativas aos ficheiros *PHP* acedidos pela *PhysioWebapp,* pela *PhysioApp* e pelo andarilho, assim como o ficheiro *connectDB.php*, que realiza a ligação à base de dados do sistema.

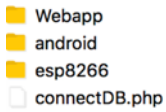


Figura 5.2 – Conteúdos da pasta raíz do servidor.

- ***Webapp*:** Esta pasta, contém os ficheiros *PHP* das páginas da *PhysioWebapp* que são apresentados na Figura 5.3. Na raíz desta pasta encontram-se a página de login (*login.php*), a de logout (*logout.php*), a de erro (*NotLoggedIn.php*), assim como uma pasta com ficheiros *PHP* que são acedidos por cada um dos tipos de utilizador (administrador, fisioterapeuta e paciente). (Estes ficheiros estão amplamente descritos na sub-secção 2.4.2 do Apêndice C).

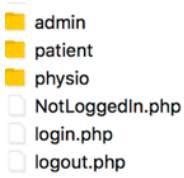


Figura 5.3 – Conteúdo da pasta *Webapp* do servidor.

A Figura 5.4, apresenta o conteúdo da pasta *admin*, que aloja os ficheiros *PHP* relativos a secção da *PhysioWebapp* relativa aos admistradores do sistema. Estes ficheiros interagem com a base de dados do sistema e permitem que os administradores realizem as ações que lhes são permitidas pelo sistema.

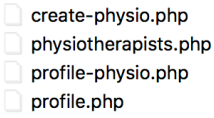


Figura 5.4 – Conteúdo da pasta *admin* dentro da pasta *Webapp* no servidor.

A Figura 5.5, por outro lado, mostra o conteúdo da pasta *patient*, que aloja os ficheiros *PHP* relacionados com secção da *PhysioWebapp* relativa aos pacientes do sistema. Estes ficheiros permitem que os pacientes vejam o seu perfil, as suas sessões e planos de exercícios.

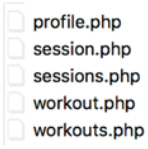


Figura 5.5 – Conteúdo da pasta *patient* dentro da pasta *Webapp* no servidor.

Finalmente, a Figura 5.6, apresenta os ficheiros *PHP* relativos à secção da *PhysioWebapp* para os fisioterapeutas, que estão incluídos na pasta *physio*. Os ficheiros *PHP* incluídos nesta pasta permitem aos fisioterapeutas realizarem as ações que lhes são permitidas, como criar pacientes ou planos de exercícios, e visualizar as sessões e planos de treino existentes.

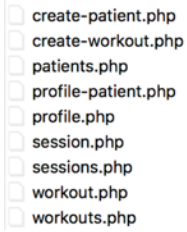


Figura 5.6 – Conteúdo da pasta *physio* dentro da pasta *Webapp* no servidor.

- **android:** Esta pasta contém todos os ficheiros *PHP*, mostrados na Figura 5.7, que permitem que todos os utilizadores, independentemente do seu tipo, possam interagir com a base de dados a partir da aplicação móvel do sistema, a *PhysioApp*. Com estes ficheiros os utilizadores poderão fazer *login*, e realizar as ações que o sistema lhes permite, recebendo ou enviando dados de/para a base de dados do sistema. (Estes ficheiros estão amplamente descritos na sub-secção 2.4.3 do Apêndice C).

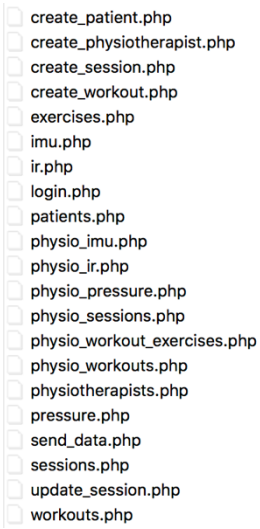


Figura 5.7 – Conteúdo da pasta *android* do servidor.

- **esp8266:** Esta pasta inclui os ficheiros *PHP* que realizam a comunicação entre o andarilho e a base de dados. Os ficheiros registam os dados relativos a sessões realizadas e as que estão em curso (ver Figura 5.8). (Estes ficheiros estão amplamente descritos na sub-secção 2.4.1 do Apêndice C).

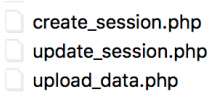


Figura 5.8 – Conteúdo da pasta *esp8266* do servidor.

5.2. Base de dados

O sistema *IoPhyR* contém uma base de dados alojada no servidor que aglomera todos os dados do mesmo. Esta é uma base de dados *MySQL* devido não só à arquitetura escolhida para o servidor, a *LAMP*, mas também devido a sua popularidade. O seu *design* foi elaborado utilizando a ferramenta *MySQL Workbech*, que permite criar bases de dados de uma maneira visual, permitindo depois alojá-la ou migrá-la num/para servidor exportando-a para um ficheiro *SQL* [70]. Ficheiro esse que ao ser importado

para a ferramenta *phpMyAdmin,* presente no servidor, permite a gestão da base de dados do sistema [71].

Esta base de dados, com o seu diagrama *EER* apresentado na Figura 5.9, contém informações pessoais dos utilizadores (administradores, fisioterapeutas e pacientes), informações relativas aos planos de exercícios criados e às sessões efectuadas pelos pacientes.

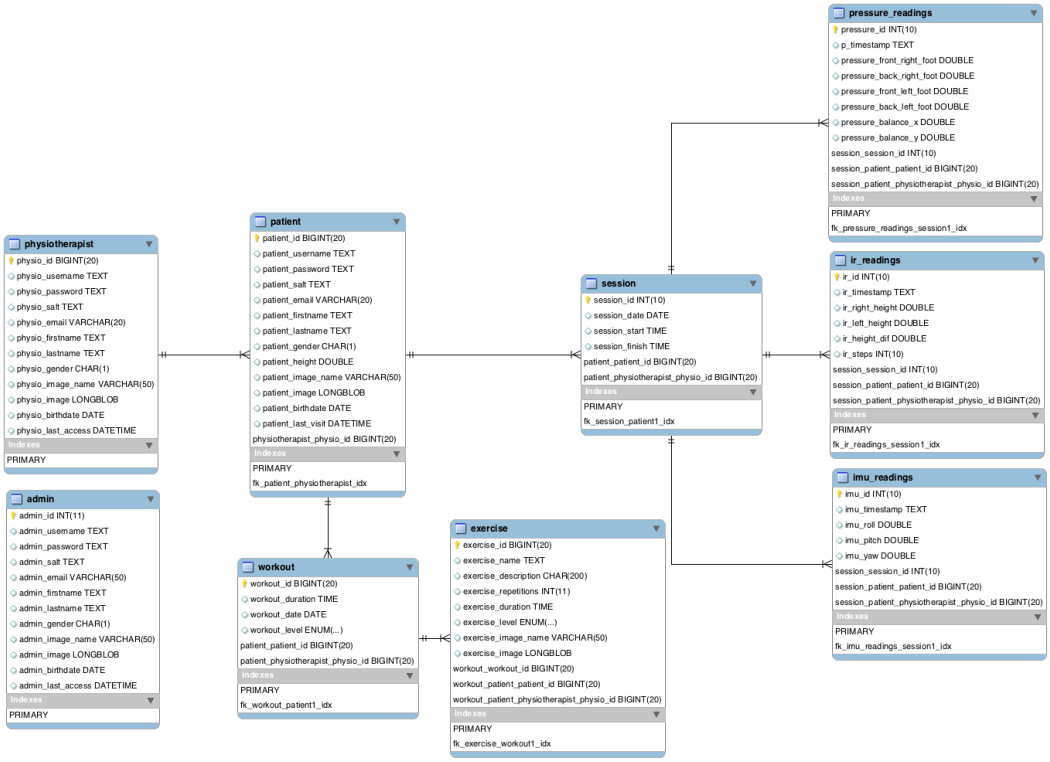


Figura 5.9 – Diagrama *EER* da base de dados do sistema.

A base de dados do sistema contém as seguintes 9 classes, como se verifica na Figura 5.9:

- ***admin*:** Esta classe guarda as informações pessoais dos administradores do sistema, como o nome, o género, a data de nascimento, a data do último *login*, o *e-mail*, assim como as credenciais necessárias para este entrar na aplicação móvel (*username* e *password*). A chave primária é o *admin_id* que é representado por um *datatype INT* de até 11 *bits*.
- ***physiotherapist*:** Esta classe guarda o mesmo tipo de informações que a classe *admin*, no entanto para os utilizadores que são fisioterapeutas. A chave primária é o *physio_id* e é representado por um *datatype BIGINT* de até 20 *bits*.

- ***patient*:** A classe *patient* guarda, o mesmo tipo de informação que as outras duas classes de utilizadores, no entanto aplicadas aos utilizadores que são pacientes. Esta classe tem uma associação de um para muitos com a classe *physiotherapist*, de modo a associar cada paciente ao fisioterapeuta correspondente. A chave primária desta classe é o *patient_id*, que é representado por um *BIGINT* de até 20 *bits*, e a chave estrangeira da classe é a chave primária da classe *physiotherapist*, o *physio_id*.
- ***workout*:** A classe *workout* guarda a informação relativa aos planos de exercícios realizados ou a realizar pelos pacientes. Entre os dados guardados estão a duração, a data, e o nível de dificuldade do mesmo. Esta classe tem uma relação de um para muitos e outra de muitos para um, com a classe *patient* e a classe *exercise*, respectivamente. A sua chave primaria é o *workout_id*, do *datatype BIGINT* de até 20 *bits*, e a sua chave estrangeira a chave primaria da classe *patient*, de modo a associar cada plano de exercícios ao paciente correspondente.
- ***exercise*:** Esta classe tem como objetivo guardar as informações relativas a cada exercício criado. Entre outras informações, a classe *exercise* guarda o nome do exercício, uma descrição, uma fotografia, a duração do mesmo, o número de repetições e o nível de dificuldade do mesmo. Cada exercício pertence a um dado plano de exercícios, ou *workout*, e desse modo existe uma relação de um para muitos desta classe com a classe *workout*, originando a chave estrangeira da classe *exercise*. Esta classe tem como chave primária o *exercise_id*, do *datatype BIGINT* com até 20 *bits*, que identifica o exercício e que garante que é único.
- ***imu_readings*:** Esta classe guarda a informação relativa a cada uma das aquisições das métricas do sensor *IMU* (*Inertial Measurement Unit*). A classe *imu_readings* contém dados como a *timestamp* de cada aquisição, os valores dos ângulos de *roll*, *pitch* e *yaw* do andarilho. Esta classe tem uma relação de "um para muitos" com a classe *session*, já que estes dados fazem parte de uma sessão de fisioterapia, realizada por um paciente sob a supervisão de um fisioterapeuta. Por fim, a classe *imu_readings* tem como chave primária o *imu_id*, um identificador representado por um inteiro de até 10 *bits*, e como chave extrangeira, a chave primária da classe *session*.

- ***ir_readings*:** Esta classe guarda os dados adquiridos dos dois sensores de ultrasons (*HC-SR04*). A *ir_readings* contém dados como a *timestamp* da leitura dos dados, o valor das elevações esquerda e direita do andarilho, e o número de passos dados até ao momento da aquisição. Esta classe tem uma relação de "um para muitos" com a classe *session* uma vez que estas medições fazem parte de uma sessão de fisioterapia. A chave primária desta classe é a *ir_id*, um identificador representado por um inteiro de até 10 *bits*, e como chave extrangeira a chave primária da classe *session*.
- ***pressure_readings*:** A classe *pressure_readings* tem como objetivo guardar as métricas adquiridas dos sensores de força aplicados em cada um dos pés do andarilho. Esta classe contém dados como a *timestamp* da leitura dos dados, os valores de força, em Newtons, exercidos em cada um dos pés do andarilho e os valores do centro de gravidade momentâneo do paciente. Existe uma relação de "um para muitos" com a classe *session*, uma vez que estes dados fazem parte de uma sessão de fisioterapia. A chave primária é o *pressure_id*, um inteiro de até 10 *bits*, e a chave estrangeira é a chave primaria da classe *session*.

Na Figura 5.10 pode ver-se parte do ficheiro *SQL* da base de dados do sistema relativa à criação de uma sessão de fisioterapia, a tabela *session*. Entre outras coisas pode-se constatar que a chave primária é *AUTO_INCREMENT*, i.e., incrementa a cada nova inserção e que é um inteiro *unsigned* (sem sinal), visto que este número só pode ser positivo e de maneira a maximizar o número de identificadores de sessão possíveis, atribuindo todos os bits da estrutura de dados *INT* aos números positivos. Pode-se constatar ainda que todos os campos, com excepção do *session_finish* são de preenchimento obrigatório (*NOT NULL*). Isto deve-se ao facto de que quando se inicia a sessão, e se regista essa mesma sessão na base de dados, não se sabe necessariamente quando irá terminar.

```
CREATE TABLE IF NOT EXISTS `session` (
  `session_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `session_date` date NOT NULL,
  `session_start` time NOT NULL,
  `session_finish` time DEFAULT NULL,
  `patient_patient_id` bigint(20) unsigned NOT NULL,
  `patient_physiotherapist_physio_id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY (`session_id`,`patient_patient_id`,`patient_physiotherapist_physio_id`),
  KEY `fk_session_patient1_idx` (`patient_patient_id`,`patient_physiotherapist_physio_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1159 ;
```

Figura 5.10 – Fragmento do ficheiro *SQL* da base de dados.

5.3. Ficheiros *PHP*

A função dos ficheiros *PHP* neste sistema, como já foi referido anterioremente, é a de realizar a interação da base de dados com os dispositivos do mesmo (com a *PhysioApp*, a *PhysioWebapp* ou o andarilho), através do protocolo *HTTP*. Os "clientes" do sistema, quando pretendem realizar uma acção, realizam pedidos *HTTP* ao servidor acedendo aos ficheiros *PHP* conveninentes, recebendo destes respostas estruturadas em objectos *JSON*.

Existem vários tipos de pedidos *HTTP* sendo os mais usuais o *GET* e o *POST*. No entanto para o sistema *IoPhyR*, devido a tratar dados sensíveis e uma vez que o pedido *GET* envia as suas variáveis directamente no endereço *URL*, foi apenas considerado o pedido *POST* apresentado abaixo:

POST

Este é um pedido que encapsula as variáveis enviadas de maneira a que estas não fiquem acessíveis. Desta maneira, os dados enviados não ficam gravados em *cache* nem no histórico dos navegadores. Por outro lado, o pedido *POST* não tem restrições em relação ao tamanho de dados enviados, o que é uma vantagem quando se pretende enviar grandes volumes de dados.

5.3.1. Clientes *PHP*

Os ficheiros PHP diferenciam-se pelos tipo clientes que a eles acedem. Nesse sentido, o sistema *IoPhyR* conta com os seguintes clientes:

- ***Webapp*:** Estes ficheiros *PHP* são usados pela *PhysioWebapp*. Os utilizadores ao entrarem na aplicação *web*, podem realizar as acções que lhes são permitidas, acedendo aos ficheiros PHP adequados que os ligam à base de dados do sistema. Estes ficheiros permitem a criação e visualização de utilizadores e planos de exercício, assim como a visualização dos dados de sessões de fisioterapia.
- ***Android*:** Estes ficheiros servem os utilizadores da *PhysioApp*. Estes utilizadores, através do acesso a estes ficheiros *PHP*, podem realizar as funções que lhes são permitidas pelo sistema. Entre essas funções destacam-se a criação ou visualização de utilizadores, planos de exercício ou sessões, assim como o envio em tempo real de dados de sessões de fisioterapia, e o término dessas mesmas sessões.

- ***Andarilho*:** Estes ficheiros *PHP* são usados pelo *smart walker*, i.e., o andarilho "inteligente". As funções destes ficheiros são as de criar uma nova sessão de fisioterapia, o envio dos dados das sessões em curso, assim como o término dessas mesmas sessões.

5.3.2. Principais ficheiros *PHP*

Cada tipo de "cliente" tem acesso a um conjunto específico de ficheiros PHP aos quais acede para interagir com a base de dados dos sistema *IoPhyR*, podendo receber ou enviar dados.

Nesse sentido, entre os ficheiros *PHP* da aplicação *web*, a *PhysioWebapp*, destacam-se os seguintes:

- ***webapp/physio/create_patient.php:*** Este ficheiro permite aos fisioterapeutas criarem um novo paciente mediante o preenchimento de um formulário *HTTP POST* com os dados do novo paciente. Após o registo bem sucedido, o fisioterapeuta que criou o paciente fica responsável por criar planos de exercício e avaliar os dados das sessões desse paciente.
- ***webapp/physio/workouts.php:*** Este ficheiro permite que os fisioterapeutas listem, por ordem cronológica, os planos de exercício de um determinado paciente que esteja a seu cargo. Para isso é necessário realizar um pedido *HTTP POST* com o *id* do paciente pretendido na variável *patient_id*. Após escolhido o plano de exercício que se pretende ver, é usado o ficheiro *workout.php* para mostrar os dados do plano de exercícios escolhido, utilizando para isso o *workout_id*, o identificador do plano.
- ***webapp/physio/session.php:*** Com este ficheiro o fisioterapeuta pode aceder aos dados de uma sessão específica de um paciente. Ao escolher-se a sessão que se pretende ver, dentre as presentes numa lista, é usado o identificador desta, o *session_id*, para aceder aos dados da mesma, e mostrá-los ao fisioterapeuta.

Em relação aos ficheiros *PHP* acedidos pelos utilizadores da aplicação móvel realçam-se os seguintes:

- ***android/patients.php:*** Este ficheiro permite que os fisioterapeutas tenham no seu dispositivo móvel, com a aplicação móvel do sistema, os dados de todos os seus pacientes. Os dados dos pacientes são depois armazenados numa base de dados local, no dispositivo, após serem recebidos num *JSON Object Array*.

- ***android/create_session.php:*** Com este ficheiro, os fisioterapeutas registam o início de uma sessão de fisioterapia. Este ficheiro é acedido quando se pretende dar início a uma sessão de fisioterapia a partir de um dispositivo móvel com a *PhysioApp*.
- ***android/send_data.php:*** Este ficheiro permite enviar os dados de uma sessão em curso para a base de dados. Os dados enviados pelo andarilho e recebidos no dispositivo móvel com a *PhysioApp*, são enviados num pedido *HTTP POST* juntamente com os identificadores do paciente e do fisioterapeuta para efeitos de organização e de autenticação.
- ***android/update_session.php:*** Este ficheiro tem a função de registar a hora de término da sessão de fisioterapia na base de dados. O ficheiro recebe os identificadores do paciente em questão e do seu fisioterapeuta e pesquisa a última sessão deste para registar a hora do término desta.

Por fim, os ficheiros *PHP* que servem o andarilho são os seguintes:

- ***esp8266/create_session.php:*** Este ficheiro encarrega-se de criar uma nova sessão na presente data para o paciente cujo *id* do cartão foi lido pelo andarilho ao iniciar a prática.
- ***esp8266/upload_data.php:*** Este ficheiro tem como objectivo inserir os métricas calculadas durante a sessão na base de dados do sistema. O ficheiro recebe o identificador do paciente num pedido *HTTP POST*, e através desse identificador, localiza a sessão que está a ser realizada por esse mesmo paciente e insere os dados pretendidos na base de dados.
- ***esp8266/update_session.php:*** Este ficheiro atualiza a hora do término da sessão na base de dados. O ficheiro recebe num pedido *HTTP POST* o *id* do paciente em causa, e após localizar a sessão em questão, usando esse mesmo *id*, atualiza a hora de término da sessão de fisioterapia.

A Figura 5.11 apresenta o ficheiro *esp8266/create_session.php*, acedido pelo andarilho, para que se possa ver em pormenor a interação deste com a base de dados.

```php
<?php
    /*Check used HTTP method and the existance of the patient_id variable*/
    if($_SERVER['REQUEST_METHOD'] == 'POST' AND isset($_POST['patient_id'])){

        /*Database connection*/
        require_once("../connectDB.php");

        /*Store POST method variable*/
        $patient_id = $_POST['patient_id'];

        /*SQL query to store physiotherapist id*/
        $query = "SELECT * FROM patient WHERE patient_id='$patient_id';";
        $resultSet = mysqli_query($con, $query);
        $row = mysqli_fetch_assoc($resultSet);
        $physio_id = $row['physiotherapist_physio_id'];

        /*Set timezone*/
        date_default_timezone_set('Europe/Lisbon');

        /*Get current date and time*/
        $date = date("Y-m-d");
        $time = date("H:i:s");

        /*SQL query to insert new session on the database*/
        $sql = "INSERT INTO session (session_date, session_start, session_finish,
            patient_patient_id, patient_physiotherapist_physio_id)
            VALUES ('$date', '$time', 'NULL','$patient_id', '$physio_id')";

        /*Check if data was sucessfully inserted*/
        if ($con->query($sql) === TRUE) {
            /*Send HTTP 200 OK response*/
            var_dump(http_response_code(200));
        } else {
            /*Send HTTP Unauthorized response*/
            var_dump(http_response_code(401));
        }
    /*The used HTTP method wasn't the required or the HTTP method doesn't have the patient_id variable*/
    } else {
        /*Send HTTP Unauthorized response*/
        var_dump(http_response_code(401));
    }
?>
```

Figura 5.11 – Ficheiro *PHP* para criação de uma sessão de fisioterapia.

O ficheiro apresentado na Figura 5.11, que tem como objectivo a criação de uma nova sessão de fisioterapia, verifica primeiramente se a comunicação *HTTP* realizada com ele mesmo foi establecida com o método *HTTP POST*, e se este transporta com ele a variável *patient_id* (linha 3). Em caso afirmativo o ficheiro prossegue com o seu curso normal, e em caso negativo envia ao dispositivo que iniciou a comunicação uma resposta *HTTP* com o código 401 *Unauthorized*.

Admitindo que o método *HTTP* usado foi o requerido (*POST*), e que a variável *$_POST['patient_id']* existe, segue-se um acesso único ao ficheiro *connectDB.php* que lida com a ligação a base de dados do sistema, e que tem acesso às credenciais de acesso à mesma (linha 6). Posteriormente é guardado na variável local *$patient_id* o identificador do paciente transportado no método *POST* (linha 9). A acção seguinte realiza a extração do *id* do fisioterapeuta, uma vez que será necessário para a criação de uma nova sessão de fisioterapia. Sendo assim, é realizada e executada uma *query SQL* para obter os dados do paciente com identificador igual ao da variável *$patient_id* (linhas 12 e 13), seguida da utilização do método *mysqli_fetch_assoc()*, de forma a estruturar o resultado dessa *query* num *array* associativo e guardá-lo na variável local *$row* (linha 14). Finalmente é guardado o *id* do fisioterapeuta em questão na variável local *$physio_id*, acedendo à posição do *array $row* relativa ao identificador do fisioterapeuta, a *$row[physiotherapist_physio_id]* (linha 15). Posteriormente é establecido o fuso horário da sessão (linha 18), e são obtidas as horas e a data atuais de

maneira a organizar cronologicamente a sessão (linhas 21 e 22). De seguida é realizada a *query* de inserção da nova sessão na base de dados com os dados previamente recolhidos (linhas 25-27), sendo depois executada e verificada a boa execução da mesma. (linhas 30-36). Finalmente, é enviada uma resposta *HTTP* para o dispositivo que efectuou o pedido, que no caso da sessão ter sido criada adopta o código *200 OK* (linha 32), e em caso desta não ter sido autorizada, o código 401 *Unauthorized* (linha 40).

5.3.3. Respostas *HTTP*

Em relação às respostas aos pedidos *HTTP*, estas são estructuradas no formato *JSON* pelos ficheiros *PHP*. Este formato permite a criação de vectores e objectos que contém pares nome-valor de forma a organizar as informações que se pretendem transmitir [72].

Na Figura 5.12, apresenta-se uma resposta a um pedido *HTTP* onde um fisioterapeuta requer o envio das sessões de fisioterapia de um dos seus pacientes. Nesse sentido, a resposta, que usa o formato *JSON*, apresenta 3 vectores com informações relativas ao pedido. No primeiro (linhas 2 a 6), envia a *String auth* que mostra se a autenticação foi bem sucedida, e o *Integer number_of_sessions* que indica o número de sessões que o paciente tem, e que foram enviadas. E os 2º e 3º vectores apresentam os dados relativos a cada uma das sessões do paciente, onde se incluem, entre outras informações, a data e hora da mesma (linha 7 a 22).

```
1  {
2    "info": [
3
4      {"auth" : "success"},
5      {"number_of_sessions" : 2}
6    ],
7    "session1": [
8      {"session_number":"1"},
9      {"session_date":"2018/06/07"},
10     {"session_start":"13:00:00"},
11     {"session_finish":"13:30:00"},
12     {"patient_number":130301290},
13     {"physiotherapist_number":12}
14   ],
15   "session2": [
16     {"session_number":"2"},
17     {"session_date":"2018/06/16"},
18     {"session_start":"16:00:00"},
19     {"session_finish":"16:30:00"},
20     {"patient_number":130301290},
21     {"physiotherapist_number":12}
22   ]
23 }
```

Figura 5.12 – Resposta *HTTP* formatada em *JSON*.

Capítulo 6 – Aplicações do utilizador

Neste capítulo pretende-se apresentar as aplicações desenvolvidas de modo a que os utilizadores do sistema possam ver os dados do mesmo. Estas são uma aplicação móvel, a *PhysioApp*, e uma *webapp*, a *PhysioWebapp*. De referir ainda que a utilização de ambas as plataformas é exclusivo para os utilizadores registados no sistema.

6.1. Definição das aplicações dos utilizadores

Durante o desenvolvimento do sistema foi indentificada a necessidade de desenvolver uma aplicação para que os utilizadores pudessem visualizar os dados das sessões ou dos planos de exercício. Esta deveria ser portátil e abrangente. Portátil devido a variabilidade dos espaços onde se realizam as sessões de fisioterapia, em casa do paciente ou nos centros de reabilitação. E abrangente de modo a estar disponível para o maior número de utentes. Nesse sentido, foi decidido o desenvolvimento de uma aplicação móvel, já que esta cumpre os requisitos determinados, podendo ser transportada pelos utilizadores em dispositivos móveis como *tablets* ou *smartphones*.
Posteriormente foi escolhido o tipo de aplicação móvel (Ver Figura 6.1) a desenvolver tendo em conta os seguintes tipos de soluções existentes:

- **Aplicações nativas.**
- **Aplicações hibridas ou multiplataforma.**

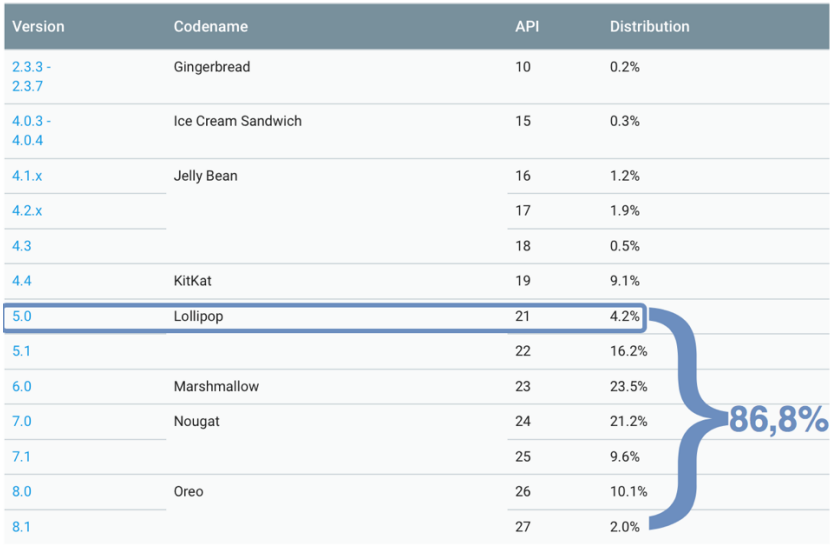
Figura 6.1 – Tipos de aplicações móveis.

A escolha do tipo de aplicação móvel recaiu nas aplicações nativas. A solução nativa apresenta-se como a melhor solução para o sistema desenvolvido, apesar do seu maior tempo de desenvolvimento quando comparada com a híbrida, devido à melhor performance, a melhor experiência de utilização e ao maior suporte dado aos desenvolvedores. Esta solução apresenta uma maior rapidez e fluidez, assim como uma melhor integração do *software* e *hardware* e um controlo do aspecto e resolução em

diferentes dispositivos que se traduzem numa melhor experiência do utilizador. Para além disso as empresas que desenvolvem os sistemas nativos dão um maior suporte aos desenvolvedores, provendo-os de uma alta gama de *API's* e padrões de *design* e *user experience* que facilitam o desenvolvimento ao programador [73], [74].

Finalmente, e escolhida a solução nativa, foi decidida a plataforma para a qual será desenvolvida e onde será instalada a aplicação do sistema. No mercado existem actualmente duas grandes plataformas, o *Android*, desenvolvido pela *Google* e o *iOS*, desenvolvido pela *Apple*. Ambas as plataformas permitem o desenvolvimento da aplicação pretendida, no entanto a escolha recaiu no *Android* por questões de orçamento e de abrangência. A primeira questão, o orçamento, tem que ver com o preço substancialmente superior dos dispositivos da *Apple* em relação aos que contém o *Android*. A escolha do *Android* permite que as clinicas reduzam gastos e/ou adquiram mais dispositivos. A segunda questão, a abrangência, tem a ver com o facto do *Android* ser o sistema operativo móvel preferido dos consumidores, tendo registado no primeiro trimestre de 2017 uma cota de mercado de 85% dos dispositivos móveis a nível mundial [75]. Este facto, faz com que haja potencialmente mais utilizadores com dispositivos compatíveis com a aplicação desenvolvida.

Para o desenvolvimento da aplicação foi utilizado o *IDE* disponibilizado pela *Google*, o *Android Studio* na versão 3.0, assim como a linguagem de programação *Java* e a linguagem de marcação *XML*. Tendo sido também definidos os requisitos mínimos para correr a aplicação, por outras palavras a versão mínima requerida do *Android*. Nesse sentido foram analisados os dados estatísticos de dispositivos activos por cada versão do *Android*, que são disponibilizados no site do sistema e apresentados na Figura 6.2. Tendo esses dados em conta, foi escolhido o *Android 5.0*, ou *Lollipop*, como versão mínima. Esta versão permite cumprir os requisitos da aplicação, acesso à internet (*Wi-Fi*, *3G* ou *4G*) e comunicação *Bluetooth*, assim como atingir uma maior abrangência, chegando a cerca de 90% dos dispositivos activos.



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.9%
4.3		18	0.5%
4.4	KitKat	19	9.1%
5.0	Lollipop	21	4.2%
5.1		22	16.2%
6.0	Marshmallow	23	23.5%
7.0	Nougat	24	21.2%
7.1		25	9.6%
8.0	Oreo	26	10.1%
8.1		27	2.0%

Figura 6.2 – Dados estatísticos da utilização das versões do *Android*.

Após a conclusão da aplicação móvel e de forma a que o sistema possa ser acedido noutros dispositivos móveis com outro sistema operativo, ou até mesmo *desktops*, foi decidido o desenvolvimento de uma *webapp* para o sistema, o *PhysioWebapp*.

PhysioWebapp

Em relação à *PhysioWebapp*, esta é desenvolvida de maneira a tornar o sistema mais abrangente, e funcional em todos os dispositivos. Esta solução apresenta-se também como uma solução de acesso prático aquando do trabalho em escritório por parte de fisioterapeutas e administradores do sistema.

Em relação às tecnologias usadas, esta interface é desenvolvida em *HTML* utilizando a *Material Design for Bootstrap 4*, uma *framework front-end open source* que permite desenvolver *websites* e *webapps* que se adaptam a diversos tipos de ecrãs e dispositivos, como ilustrado na Figura 6.3 [76]. Para além do referido, esta *framework* disponibiliza ainda *designs* semelhantes aos usados pelo sistema *Android* o que beneficia a unidade do sistema e a *user experience* dos seus utilizadores. É ainda usada a linguagem *PHP* para realizar as interações da mesma com a base de dados do sistema.

Figura 6.3 – Ilustação da utilização do *Material Design for Bootstrap 4*.

6.2. Utilizadores e fluxo das aplicações

As aplicações desenvolvidas podem ser acedidas por pelos três tipos de utilizadores do sistema, no entanto cada um deles tem acesso a diferentes informações e pode realizar diferentes acções. O administrador do sistema pode proceder inserir novos fisioterapeutas, e ver os seus dados. Por outro lado, o fisioterapeuta pode inserir novos pacientes e gerir as suas sessões e planos de treino. Por último, o paciente, pode apenas ver os seus próprios dados.

A Figura 6.4 apresenta um diagrama de actividades do administrador onde se pode ver não só as funcionalidades que este tem acesso, como a sequência de acções para as realizar.

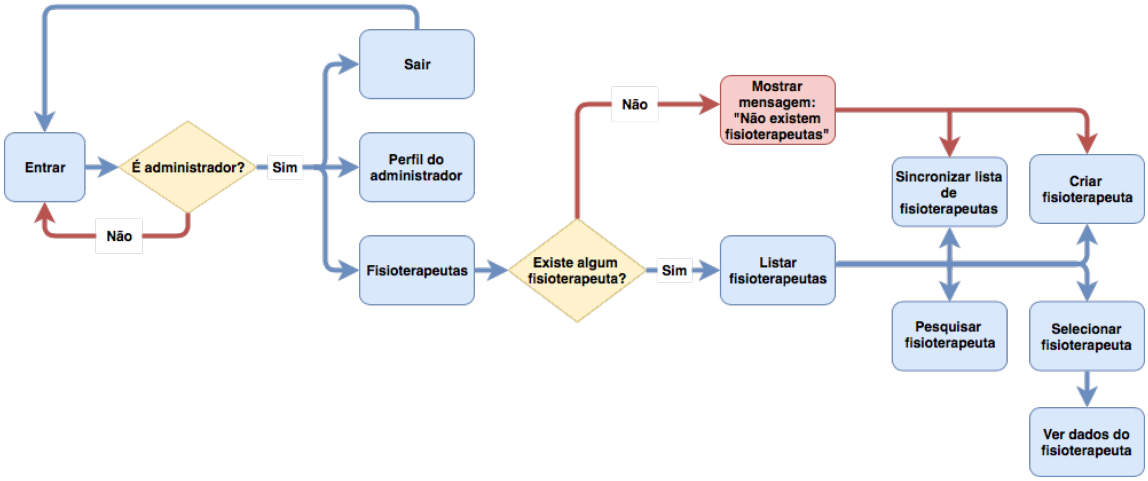


Figura 6.4 – Diagrama das funcionalidade e acções do administrador.

Após proceder ao correcto *login* do administrador este é deslocado para os seu perfil, podendo realizar duas ações, delocar-se ao menu dos fisioterapeutas ou sair da aplicação.

Entrando no menu dos fisioterapeutas, estes são listados se existirem. Caso existam, o administrador poderá pesquisar, criar ou selecionar um fisioterapeuta, para além de poder sincronizá-los com o servidor. Por outro lado, caso não exista ainda nenhum fisioterapeuta, é mostrada uma mensagem de erro, reveleando não haver qualquer fisioterapeuta no sistema. Neste caso o administrador poderá apenas criar fisioterapeutas e sincronizá-los com a base de dados do sistema.

A Figura 6.5 apresenta o diagrama de actividades do fisioterapeuta onde se pode ver as funcionalidades e a sequência das acções para as realizar.

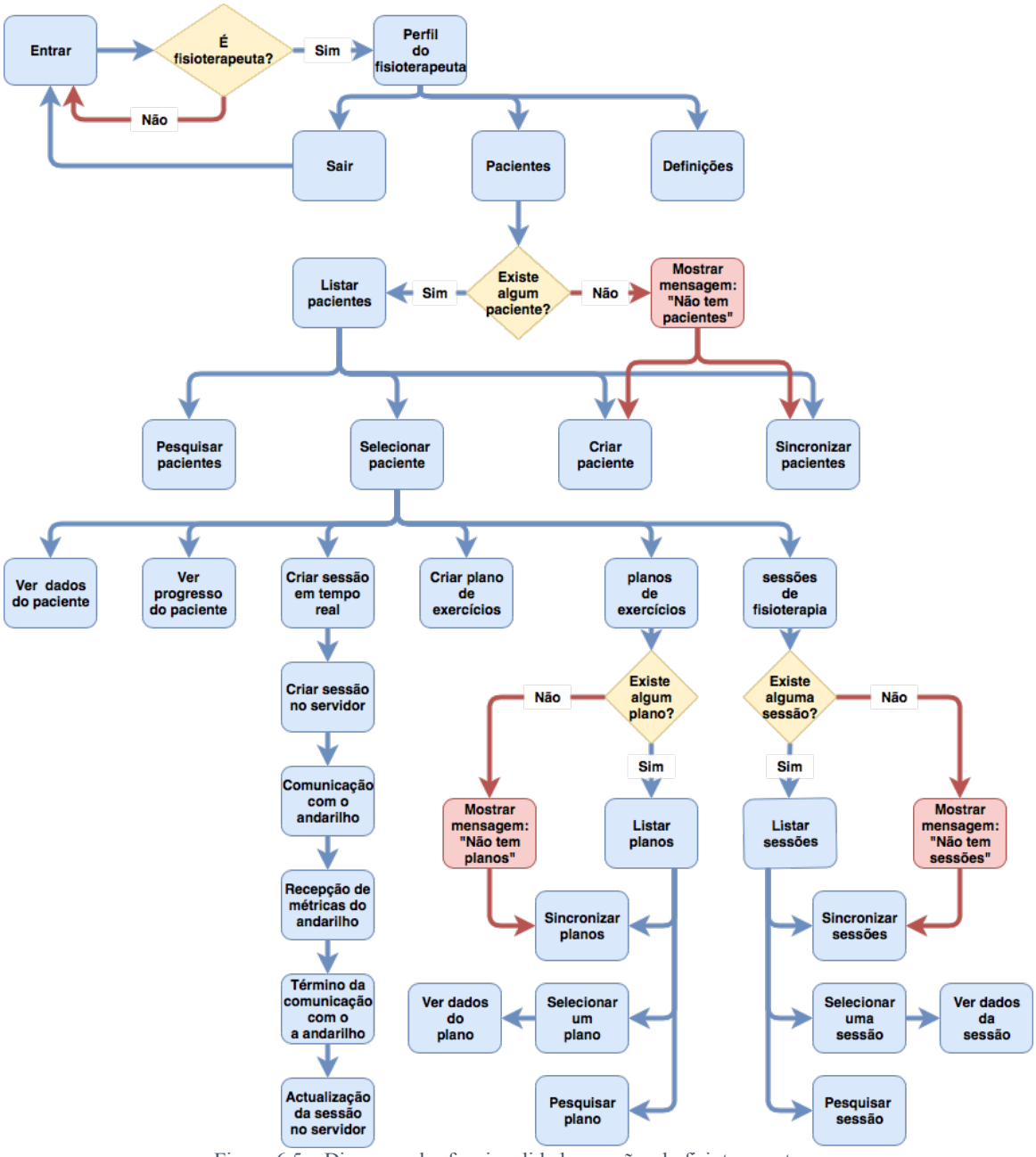


Figura 6.5 – Diagrama das funcionalidade e acções do fisioterapeuta.

Neste caso, após entrar nas aplicações do sistema, o fisioterapeuta é encaminhado para o seu perfil. Lá poderá ver os seus dados pessoais, e terá acesso a um menu lateral onde poderá sair da aplicação, aceder as definições ou aceder à área dos pacientes. Se pretender entrar nas definições, o fisioterapeuta poderá usar o *Bluetooth* do dispositivo para emparelhar o andarilho com o seu dispositivo móvel. Se por outro lado entrar na área dos pacientes, estes serão listados caso existam. Caso contrário é mostrada uma mensagem a referir esse facto, e o fisioterapeuta poderá apenas criar pacientes e sicronizá-los com o servidor do sistema. Se existir algum paciente, o fisioterepeuta poderá também pesquisá-los pelo seu nome ou e-mail ou selecioná-los e aceder ao seu perfil. Chegando ao perfil do paciente, o fisioterapeuta pode ver os dados pessoais do mesmo, assim como ver ou criar planos de exercício, ver as sessões ou criar e visualizar dados de uma sessão em tempo real, assim como ver estatísticas relativas às últimas cinco sessões do paciente. A secção dos planos de exercícios e das sessões de fisioterapia, apesar de conterem informações diferentes tem um funcionamento semelhante, como se pode ver na Figura 5.5. Por outro lado a criação de sessões em tempo real, realiza a comunicação *Bluetooth* com o andarilho recebendo métricas, mostrando-as em gráficos e enviando-as para o servidor. De referir ainda o facto de que as funcionalidades que utilizam o *Bluetooth*, e a que permite visualizar as estatísticas do paciente estão apenas disponíveis na aplicação móvel do sistema.

A Figura 6.6, por outro lado, apresenta o diagrama de actividades do paciente que expõe as funcionalidades destes, e a sequência de acções para as realizar.

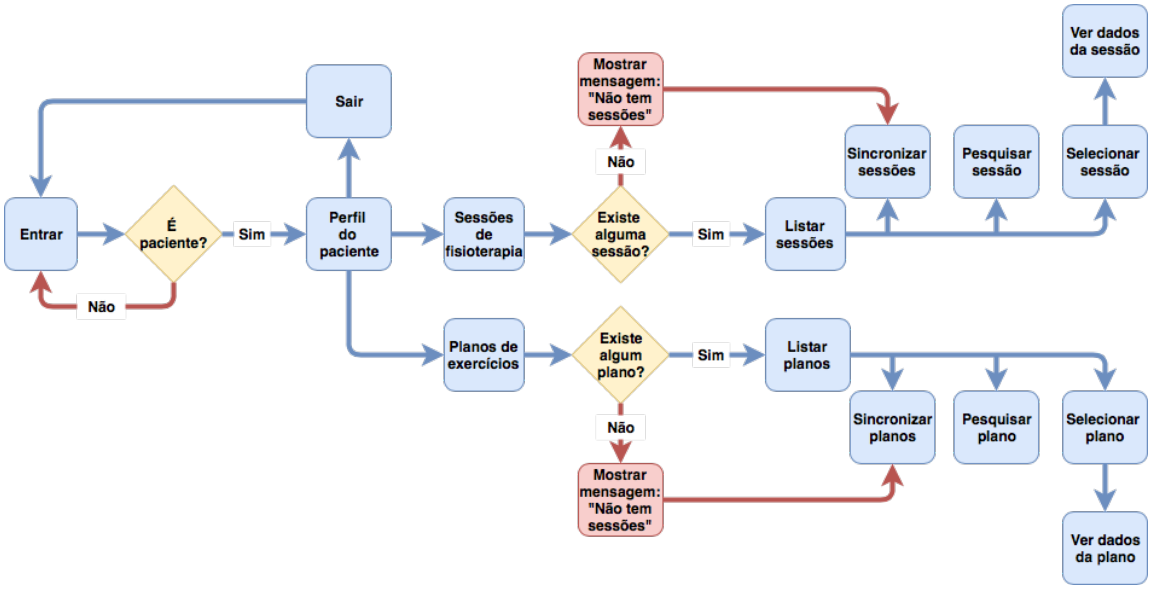


Figura 6.6 – Diagrama das funcionalidade e acções do administrador.

Neste último caso, após um login bem sucedido, o paciente entra directamente para o seu perfil. Lá poderá ver os seus dados pessoais e aceder a um menu lateral onde poderá escolher sair da aplicação, aceder à área dos planos de exercício ou das sessões realizadas. Segindo para os planos de exercício, estes serão listados caso existam, permitindo ao paciente pesquisar e ver as informações de um plano em concreto. Caso contrário, será mostrada uma mensagem a reportar a falta de planos, e o paciente poderá apenas realizar a sincronização com a base de dados para verificar se existem novos planos de exercício.

Por outro lado, caso o paciente deseje aceder à secção da área das sessões, estas ser-lhe-ão listadas caso existam. Se for esse o caso, poderá pesquisar ou aceder a alguma sessão assim como realizar a sincronização destas com o servidor. Caso não esteja disponível nenhuma sessão, é mostrada uma mensagem a referir esse facto, podendo o paciente apenas tentar sincronizar as sessões com o servidor.

6.3. Interfaces gráficas das aplicações

Nesta secção apresentam-se as interfaces gráficas, da *PhysioApp* e Physio*Webapp* desenvolvidas. A apresentação ilustra a utilização das aplicações desenvolvidas por parte de um fisioterapeuta, utilizando *printscreens*.

O fisioterapeuta quando acede às aplicações do sistema depara-se com as páginas de login apresesentadas na Figura 6.7. Nas *textfields* mostradas, insere as suas credenciais e pressiona o botão *login* para entrar no sistema.

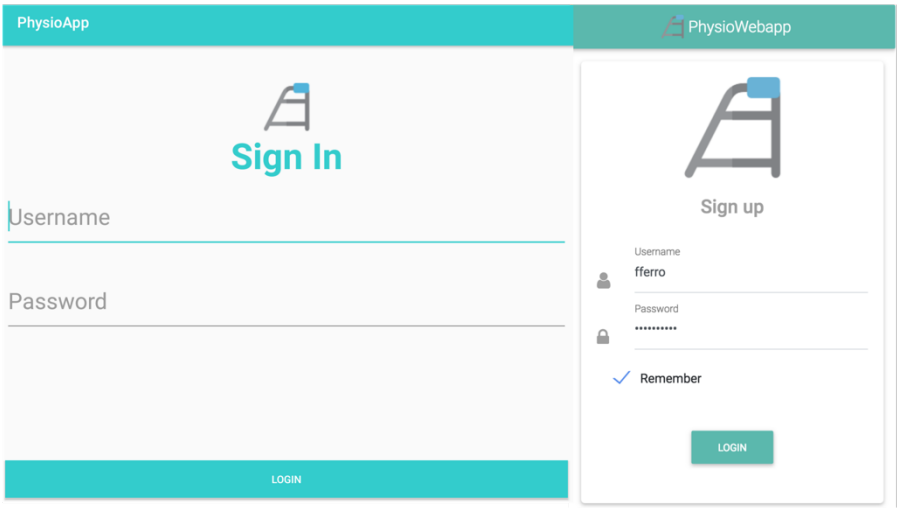


Figura 6.7 – Página de *login* da *PhysioApp* (esquerda) e da *PhysioWebapp* (direita).

Caso o *login* seja bem sucedido, é direcionado para o seu perfil que corresponde a um dos sub-menus, o *Profile*.

Sub-menu *Profile*

Neste sub-menu é possível ver os dados pessoais do utilizador, neste caso o fisioterapeuta, assim como aceder a um menu lateral, onde pode aceder novamente ao seu perfil (sub-menu *Profile*), ir para a página dos pacientes (sub-menu *Patients*), das definições (sub-menu *settings*) e sair do sistema (botão *Logout*). A página das definições está apenas disponível na aplicação móvel (Ver Figura 6.8).

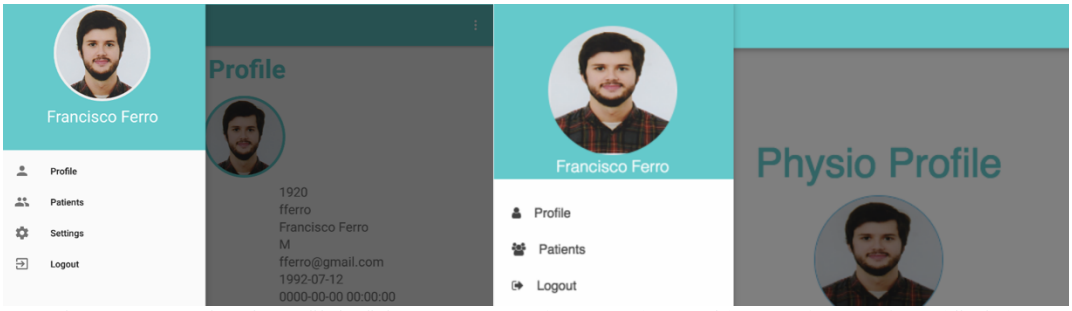


Figura 6.8 – Página de perfil do fisioterapeuta na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Sub-menu *Patients*

No menu *Patients*, o fisioterapeuta vê a sua lista de pacientes podendo actualizá-la sincronizando-a com o sistema (botão no canto superior esquerdo), criar um novo paciente (botão no canto superior direito) ou aceder ao perfil de um paciente (clicando sobre o mesmo, na lista) (Ver Figura 6.9).

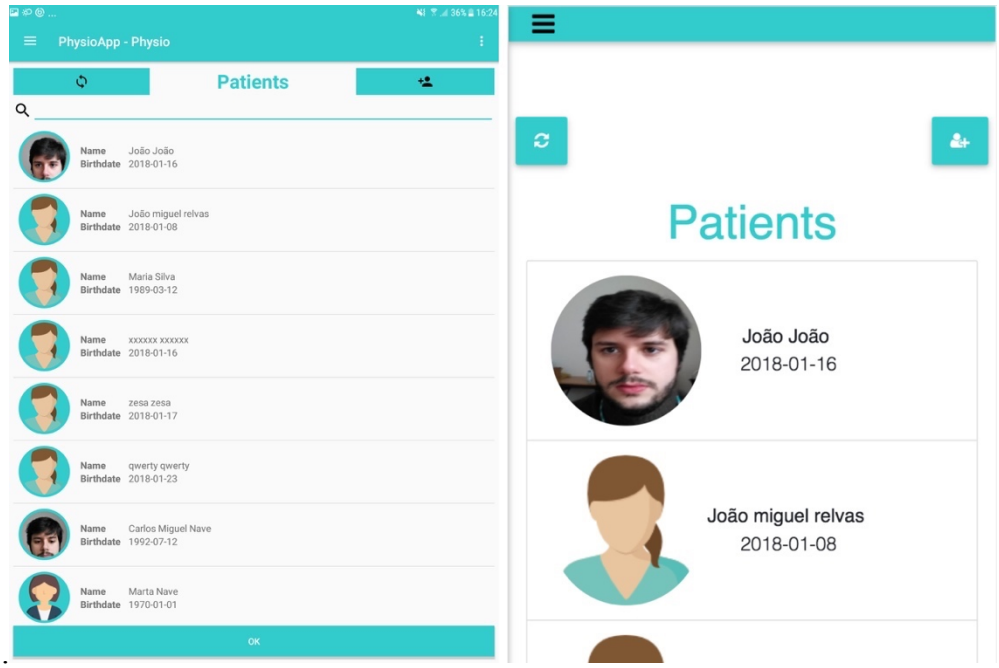


Figura 6.9 – Página do menu *Patients* na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Clicando no botão criação de paciente, é-lhe apresentada a página da Figura 6.10. Inserindo os dados pessoais, uma foto, e clicando no botão *register* procede-se ao registo do paciente. O registo é bem sucedido quando são preenchidos todos os dados e

o *e-mail*, o *username* e o *id* inseridos, não existem no sistema. A inserção do *id* no *webapp* é manual, já na aplicação móvel, é feita através da comunicação *Bluetooth* com o *PhysioRegister*, um módulo leitor de cartões *RFID*. Esta transação é realizada, usando o menu de comunicação, logo abaixo da fotografia do paciente.

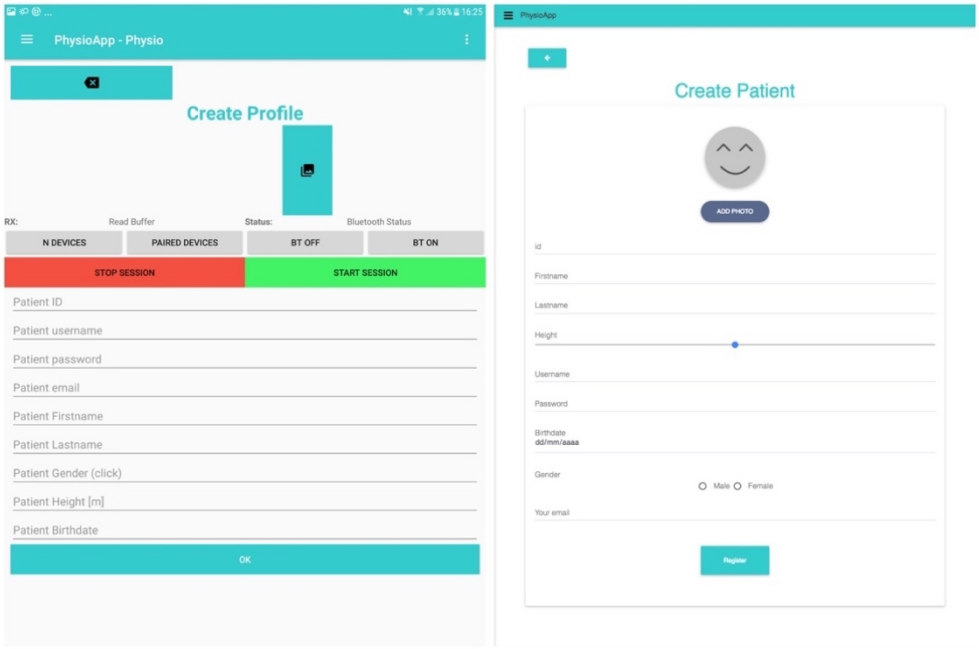


Figura 6.10 – Página de criação do paciente na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Por outro lado, clicando sobre um paciente da lista, é apresentado o perfil desse mesmo paciente (ver Figura 6.11). Nessa página pode aceder ao progresso (botão *Patient Progress*), ao histórico (botão *Patient History*) e aos planos de exercícios do paciente (botão *Patient Workouts*), assim como criar um novo plano (botão *Create Workout*), ou realizar uma sessão com visualização de dados em tempo real (botão *Real Time Session*). No entanto as funcionalidades que permitem ver o progresso do paciente e visualizar dados de sessões em tempo real são exclusivas da aplicação móvel.

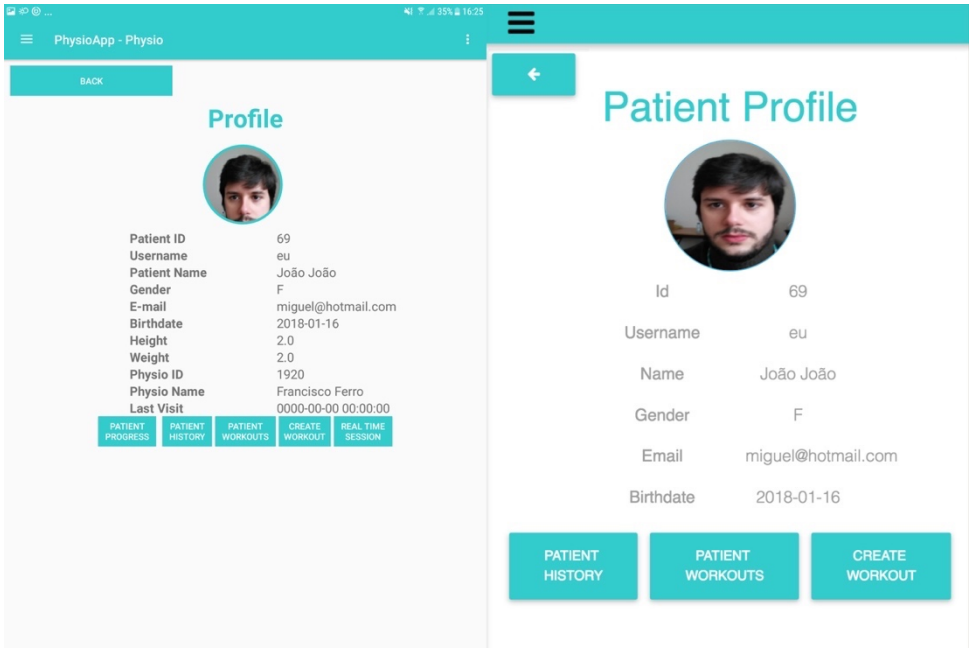


Figura 6.11 – Página de perfil do paciente na *PhysioApp* (esquerda) na *PhysioWebapp* (direita).

- **Progresso do paciente**

Dentro do perfil do paciente, e clicando no botão *Patient Progress*, acede-se a uma página onde se pode visualizar dados estatísticos da performance do mesmo. É apresentado um menu de *tabs* com gráficos com valores médios de tempo desprendido por passo, dos ângulos de rotação e da elevação e força exercida do/sobre o andarilho (ver Figura 6.12).

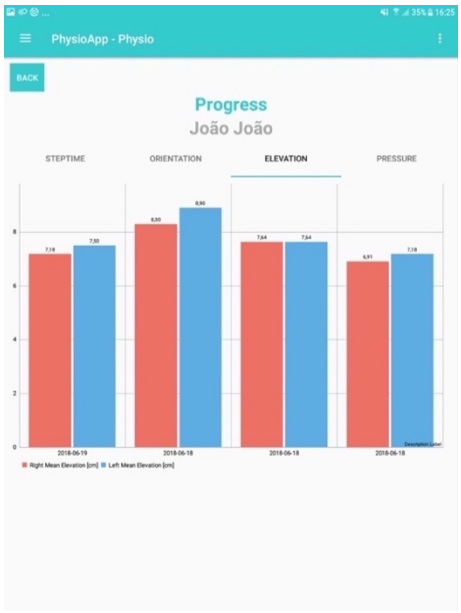


Figura 6.12 – Página de progresso do paciente na *PhysioApp*.

- **Histórico do paciente**

Clicando no botão *Patient History*, dentro do perfil do paciente, é possível ver uma lista com todas as sessões realizadas por esse mesmo paciente. Nessa página permitido voltar para o perfil do paciente (botão *Back*), actualizar a lista de sessões sincronizando-a com o sistema (botão com símbolo de sicronia), ou selecionar uma sessão, clicando sobre a mesma (ver Figura 6.13).

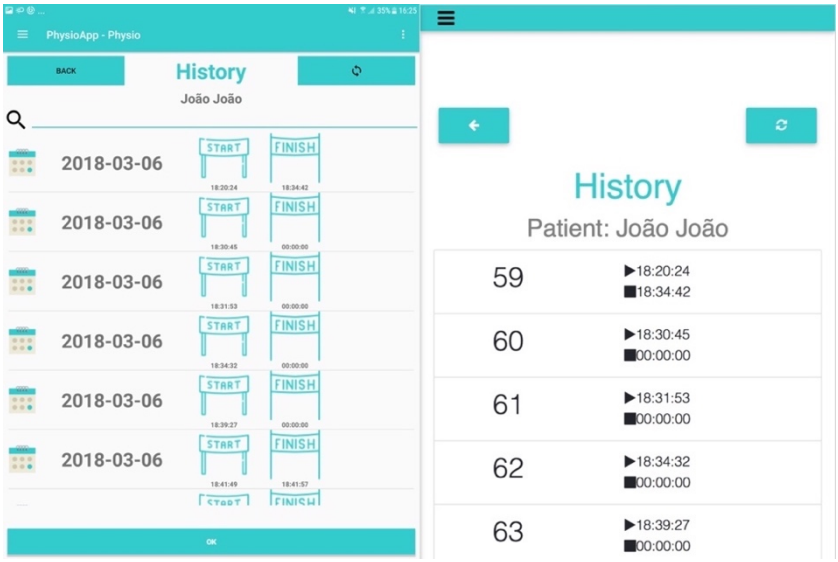


Figura 6.13 – Página do histórico das sessões de fisioterapia na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Premindo uma sessão, o fisioterapeuta pode ver informações relativas à essa mesma sessão, como o número da sessão, a data e a hora de início e fim da mesma, assim como métricas relativas à elevação do andarilho, o número de passos (ver Figura 6.14). O fisioterapeuta pode igualmente navegar no menu de tabs onde se apresentam, em forma de gráficos, os dados de orientação, elevação bilateral, de equilíbrio e força exercida sobre os pés do andarilho adquiridos ao longo da sessão. Para sair da sessão, basta premir o botão *back*, no canto superior esquerdo do ecrã.

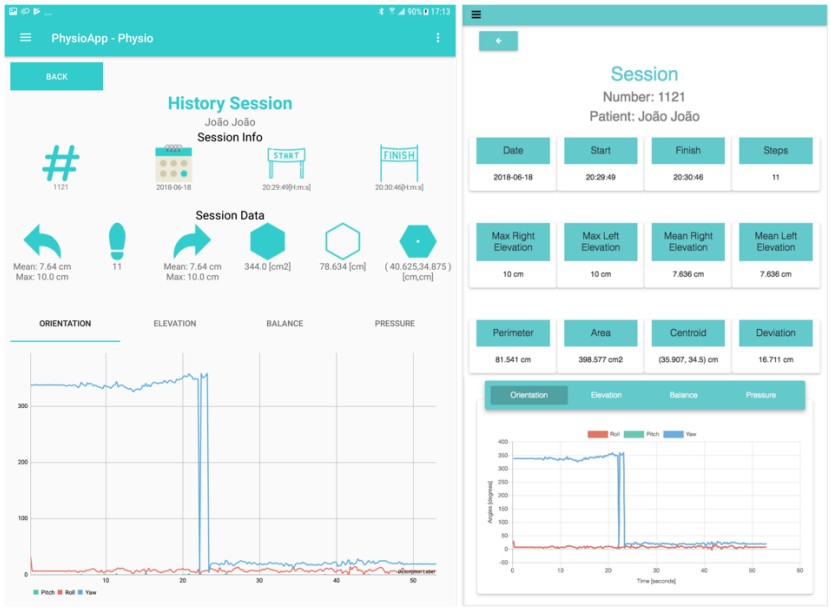


Figura 6.14 – Página de uma sessão de fisioterapia na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

- **Planos de exercícios do paciente**

Clicando no botão *Patient Workouts,* dentro do perfil do paciente, encontra-se uma lista com os planos de exercícios desse mesmo paciente. O fisioterapeuta pode voltar para o perfil do paciente (botão *Back*), actualizar os planos (botão com símbolo de sicronia), ou selecionar um plano, clicando sobre o mesmo (ver Figura 6.15).

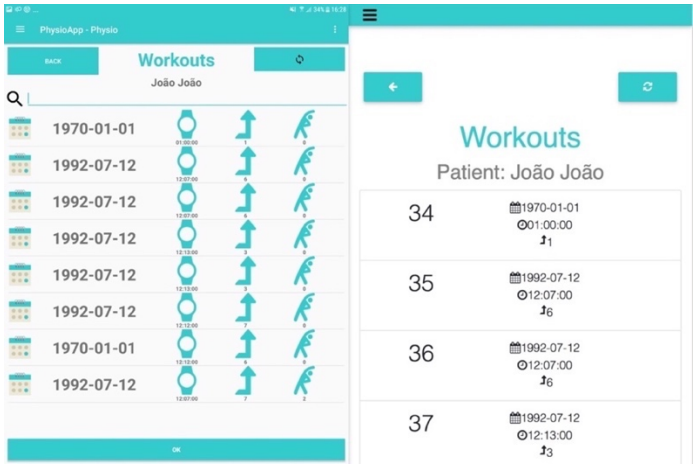


Figura 6.15 – Página do histórico de planos de exercícios na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Clicando sobre um plano de exercícios, o fisioterapeuta acede a informações relativas a essa mesma sessão como a data, hora e nível de dificuldade da mesma, e outras relativas a cada exercício incluído nesse plano (fotografia, título, repetições, duração, dificuldade e descrição). Clicando no botão *back*, regressa-se à lista de planos do paciente (ver Figura 6.16).

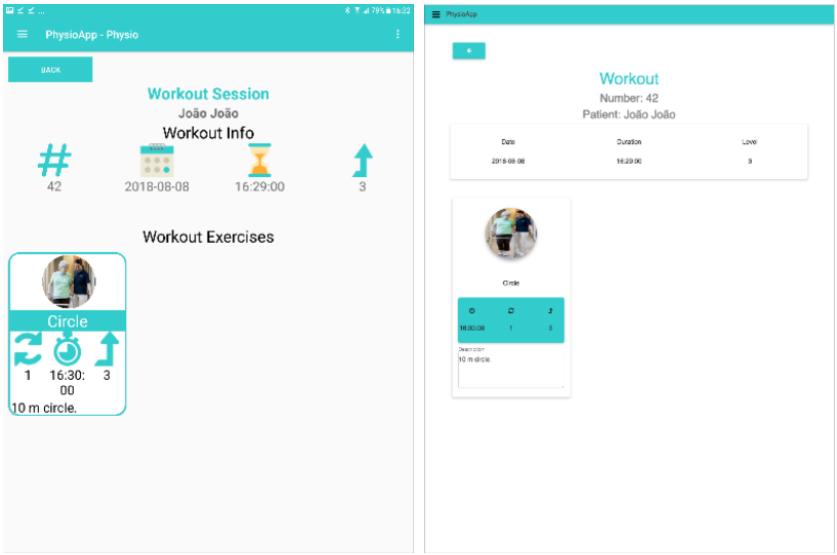


Figura 6.16 – Página de um plano de exercícios na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

- **Criar um plano de exercícios**

Clicando no botão *Create Workout*, no perfil do paciente, o fisioterapeuta pode criar um novo plano de exercícios para o paciente em questão (ver Figura 6.17). Existem dados gerais (data, duração e dificuldade), e outros relativos a cada exercício (foto, título, número de repetições, duração, nível de dificuldade e descrição).

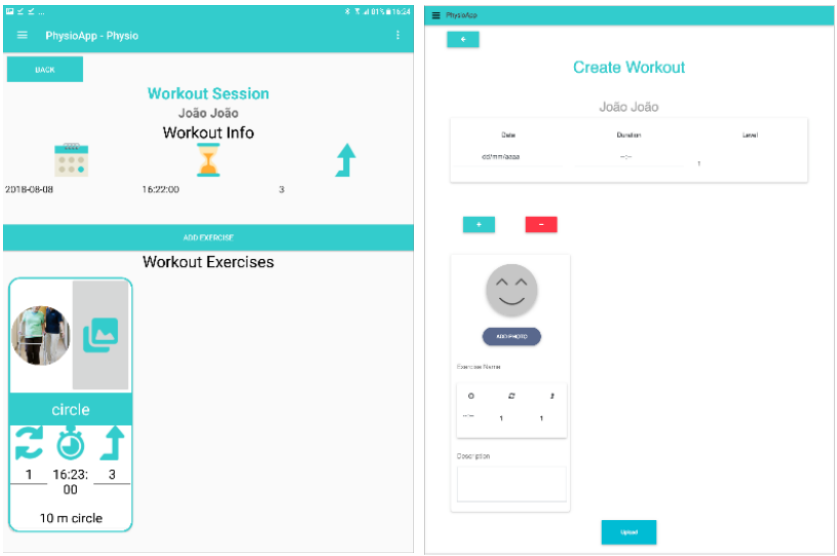


Figura 6.17 – Página de criação de um plano de exercícios na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

- **Realizar uma sessão em tempo real**

Clicando no botão *Real Time Session*, no perfil do paciente, é possível iniciar uma sessão em tempo real para esse mesmo paciente. Utilizando o painel de botões da comunicação *Bluetooth* é iniciada a transmissão de métricas do o andarilho para o dispositivo móvel. As métricas são prontamente mostradas em forma de gráficos, como se pode ver na Figura 6.18. O fisioterapeuta pode ainda regressar ao perfil do paciente clicando no botão *back*, situado no canto superior esquerdo do ecrã.

Figura 6.18 – Página das sessões em tempo real na *PhysioApp*.

Sub-menu *Settings*

Neste sub-menu, o fisiotereuta pode utilizar o *Bluetooth* do dispositivo móvel para realizar a busca e o emparelhamento com o andarilho desenvolvido (ver Figura 6.19). Para isso deve utilizar o menu de comunicação presente neste sub-menu.

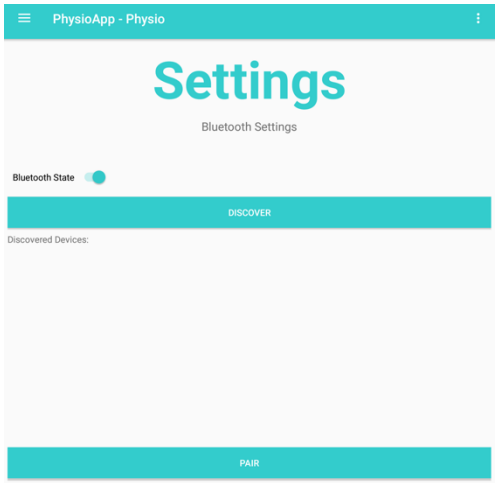


Figura 6.19 – Página das definições na *PhysioApp*.

6.4. Características fundamentais das aplicações

As aplicações desenvolvidas são constituídas pelas seguintes características:

Login nas interfaces: O administrador, fisioterapeuta ou paciente que queira utilizar as aplicações do sistema tem de estar registado no mesmo. O registo dos mesmos é diferente dependendo do tipo de utilizador. O administrador é registado manualmente na base de dados do sistema. O fisioterapeuta é registado pelo administrador, quando este utiliza as aplicações do sistema. Já o registo do paciente é realizado pelo fisioterepeuta

ao usar a *PhysioApp* ou a *PhysioWebapp*. Para realizar o *login* nas aplicações do sistema, os utilizadores registados devem inserir as suas credenciais (*username* e *password*) como mostrado na Figura 6.7.

Armazenamento local: De modo a não depender do acesso à internet para visualizar dados vistos anteriormente, a *PhysioApp* armazena numa base de dados local os dados anteriormente acedidos. É usada uma base de dados *SQLite*, que é *open source,* mais leve que a alojada no servidor, que suporta *SQL*, e usada em multiplataformas, sendo ideal para plataformas móveis (ver Figura 6.20) [77]–[79]. Este tipo de bases de dados contém apenas 5 classes (*NULL*, *INTEGER*, *REAL*, *TEXT*, *BLOB*), que englobam grande parte dos dados existentes numa base de dados *MySQL* [80]. De modo a criar e interagir com esta base de dados local é utilizada a classe *SQLiteOpenHelper*, disponibilizada no *SDK* do *Android*.

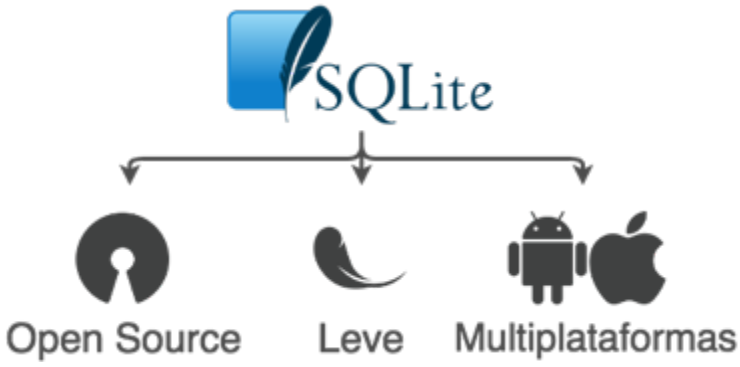


Figura 6.20 – Características do *SQLite*.

Comunicação *HTTP*: De modo a interagir com a base de dados do servidor do sistema, a *PhysioApp* e a *PhysioWebapp* utilizam pedidos *HTTP* que acedem a ficheiros *PHP* alojados no servidor para o efeito. No caso da aplicação móvel é usada a biblioteca *Volley* [81] disponibilizada pela equipa do Android. No caso da *webapp* são realizados directamente em formulários *HTML*.

Comunicação *Bluetooth*: Para realizar as sessões de fisioterapia com a visualização de dados em tempo real e para ler os cartões *RFID* para o registo de pacientes, é usada a comunicação *Bluetooth*. Nesse sentido são utilizadas as classes *BluetoothAdapter*, *BluetoothDevice*, *BluetoothSocket* e *BroadcastReceiver* do *Android SDK*. A primeira permite o acesso e uso do módulo *Bluetooth* do dispositivo móvel para começar a descoberta de dispositivos, e listagem de dispositivos emparelhados. A segunda representa o dispositivo com o qual se pretende comunicar e possibilita a ligação bluetooth. A terceira é responsável pela gestão da comunicação bidirecional com os dispositivos (andarilho e módulo leitor de cartões) através de um *InputStream* e um

OutputStream. Finalmente, a quarta permite realizar realizar operações com o módulo *Bluetooth* do dispositivo móvel, através de *Intents*, como ligar e desligar o *Bluetooth*, realizar uma descoberta de novos dispositivos, ver os dispositivos já emparelhados, ou realizar uma comunicação *Bluetooth*.

Desenho dos gráficos: Cada uma das aplicações dos utilizadores utiliza uma biblioteca para elaborar os diversos gráficos que são disponibilizados aos fisioterapeutas e pacientes (Ver Figura 6.21). Na *PhysioApp* é utilizada a biblioteca *MPAndroidChart* [82]. Esta permite desenhar diversos tipos de gráficos, entre eles os usados na aplicação desenvolvida (gráficos de linhas, de área, de barras e de dispersão). Esta biblioteca permite ainda realizar operações de *zoom* dentro dos gráficos. Já na *PhysioWebapp* é usada a biblioteca *Chart.js* para os mesmos efeitos [83].

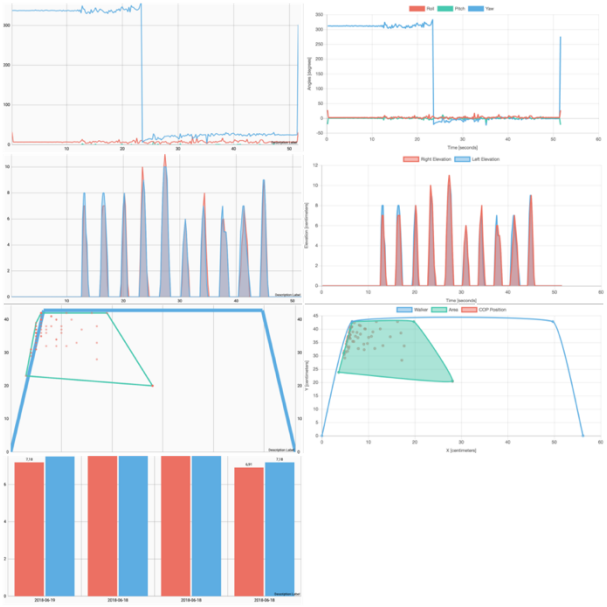


Figura 6.21 – Tipos de gráficos utilizados na *PhysioApp* (esquerda) e na *PhysioWebapp* (direita).

Capítulo 7 – Resultados

O presente capítulo pretende apresentar e demonstrar os resultados obtidos durante as duas sessões de ensaios realizadas. Sessões essas que foram objeto da elaboração de dois artigos científicos que estão disponíveis no Apêndice A. O primeiro relativo ao 1º ensaio, tendo sido aceite e apresentado na 1ª edição do *ISSI* (*International Symposium in Sensing and Instrumentation in IoT Era*) em Shangai, na China. E o 2º, referente ao 2º ensaio, que foi aceite e que será apresentado em Dezembro de 2018 na 12ª edição do *ICST* (*International Conference on Sensing Technology*), em Limerick, na Irlanda.
Devido a questões de índole moral e ética, os referidos ensaios foram realizados por grupos de voluntários saudáveis. Os dados recolhidos são relativos ao equilíbrio dos voluntários (centro de forças), à elevação bilateral do andarilho por parte dos mesmos, e aos ângulos de orientação que estes realizam durante os ensaios. De referir ainda que devido à solução desenvolvida se tratar de uma proposta de sistema de reabilitação física, os ensaios realizados tiveram como foco a validação e consistência dos dados obtidos nas mesmos, idependentemente da sua repetição.

7.1. Primeiro ensaio

Para este primeiro ensaio foi requisitado um conjunto de 5 voluntários. Voluntários esses saudáveis, do sexo masculino, com idades compreendidas entre os 24-30 anos, pesos entre os 65-80 kg, e alturas entre 1,70-1.80 metros. No que diz respeito à execução do ensaio, três dos voluntários realizaram-no simulando ser pacientes com limitações de movimentos, e os outros dois realizaram-no de forma normal. As limitações simuladas foram a afectação e comprometimento do lado direito do corpo (dois voluntários), e do lado esquerdo (um voluntário) devido a AVCs. Esses comprometimentos, teoricamente, afectariam a elevação do andarilho por parte do voluntário, elevando um lado mais do que o outro, e o seu equilíbrio, causando um desvio do centro de forças no sentido do lado afetado.

Em relação ao trajeto desenhado e implementado pelos voluntários, este consistiu numa caminhada de 10 passos em linha recta, como se pode ver na Figura 7.1. Sendo que para este ensaio, foram analisados os dados de equilíbrio do voluntário assim como a elevação do andarilho por si executada.

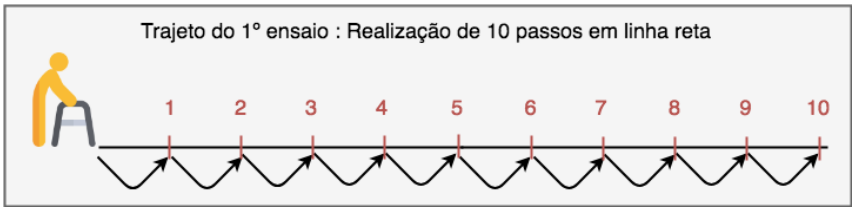


Figura 7.1 – Trajeto do 1º ensaio.

7.1.1. Resultados de equilíbrio

Em relação aos resultados de equilíbrio, as Figuras 7.2, 7.3 e 7.4 apresentam cada uma 4 padrões de equilíbrio adquiridos durante o ensaio. Nos gráficos presentes nessas figuras pode ver-se uma curva azul, que representa o andarilho visto de cima, assim como conjuntos de pontos vermelhos que simbolizam os pontos de equilíbrio instantâneo relativos à aplicação de força no andarilho.

A Figura 7.2 exibe quatro tipos de desvio de equilíbrio realizados pelo voluntário que simulou a afectação do lado esquerdo do seu corpo. Como se pode ver nesta figura, o comprometimento realizado traduz-se numa maior pressão exercida sobre o lado esquerdo do andarilho. Sendo que essa maior pressão resulta também num desvio do equilíbrio para esse mesmo lado. Pode ver-se ainda na figura apresentada que existem dois gráficos onde a condição física parece mais severa que nos outros. Os gráficos referidos são os do canto superior direito, e inferior esquerdo da Figura 7.2, e a sua severidade deve-se ao facto de terem o conjunto de pontos de equilíbrio (centro de forças) mais concentrado sobre o pé frontal esquerdo do andarilho. Por outro lado, nos restantes gráficos, pode ver-se o referido conjunto mais disperso, o que leva à percepção de uma melhor condição física.

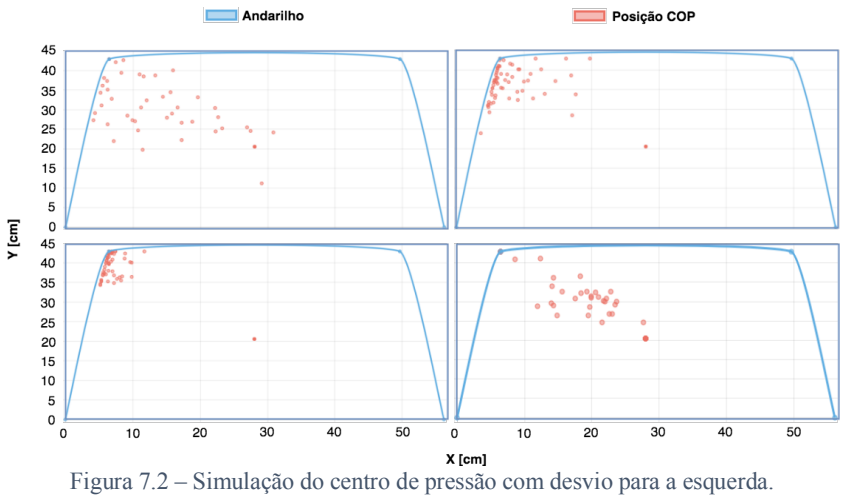


Figura 7.2 – Simulação do centro de pressão com desvio para a esquerda.

A Figura 7.3 apresenta 4 casos de equilíbrio anómalo, realizados pelos dois voluntários que simularam o comprometimento do lado direito do seu corpo. Esta figura confirma, à semelhança da anterior, um dos pressupostos iniciais. Isto é, que ter afectado um dos lados do corpo conduz a uma maior pressão e um desvio dos pontos de equilíbrio (centro de forças) para esse mesmo lado, neste caso para a direita. Para além disso, encontra-se também um caso significativamente mais severo que os outros, no canto superior esquerdo. No gráfico desse caso, o agravamento é visível dada a maior concentração dos pontos de equilíbrio existentes sobre o lado direito da estrutura do andarilho. Nos outros casos verificam-se, por contraposição, resultados menos agudos devido à maior dispersão e afastamento dos pontos referidos da estrutura do andarilho.

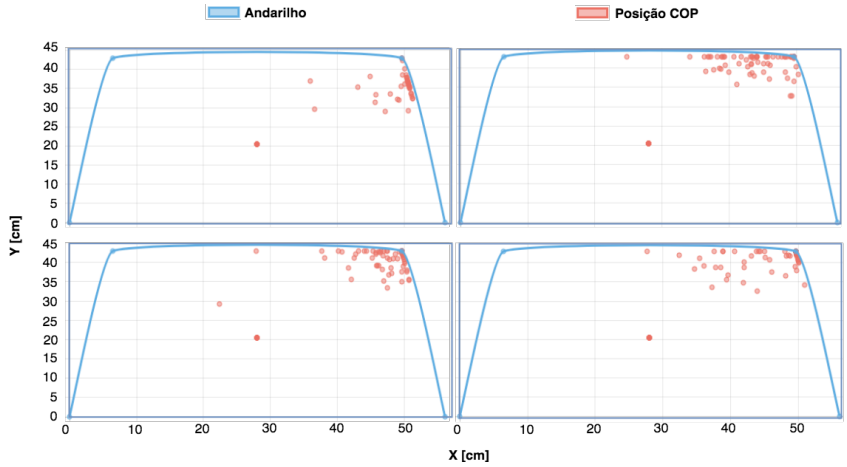


Figura 7.3 – Simulação do centro de pressão com desvio para a direita.

Nesta 3ª figura relativa ao equilíbrio, a Figura 7.4, apresentam-se quatro casos obtidos pelos dois voluntários que realizaram o ensaio normalmente. Nos 4 casos apresentados nesta figura, identifica-se um denominador comum. Sendo ele o facto de terem o conjunto de pontos de equilíbrio mais centrado e com mais dispersão. Finalmente, constata-se que ainda estes estejam dispersos, estão igualmente desviados da estrutura do andarilho.

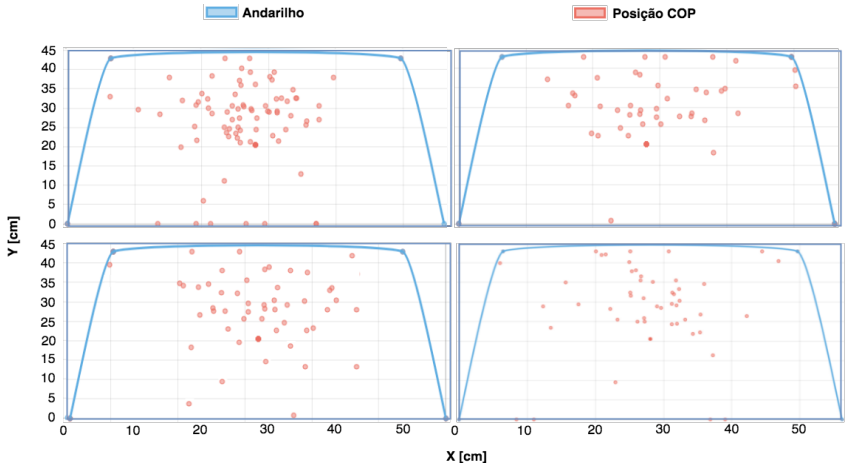


Figura 7.4 – Simulação normal do centro de pressão.

7.1.2. Resultados de elevação

Nos gráficos de elevação presentes na Figura 7.5 encontam-se duas curvas, de cores azul e vermelho. A curva azul representa a elevação do lado esquerdo do andarilho, e a vermelha a do lado direito. A elevação é medida em centímetros, e o tempo decorrido em segundos.

A Figura 7.5 apresenta dois gráficos de elevação lateral do andarilho com a finalidade de comparar a performance dum voluntário que manifesta a afectação do lado direito do seu corpo devido a um AVC (gráfico de cima), com um que não o faz (gráfico de baixo). Nesse sentido, pode verificar-se que existe uma baixa variação entre as elevações laterais do andarilho, no caso do voluntário sem limitações. Por contraposição, no caso do voluntário que simula a referida afectação, constata-se uma grande predominância na elevação do lado esquerdo do andarilho, lado contrário ao da sua limitação. Podendo esta predominância ser atribuída à falta de força no lado afectado. De referir ainda que se regista, neste caso, uma diferença de até mais 6 centímetros do lado esquerdo.

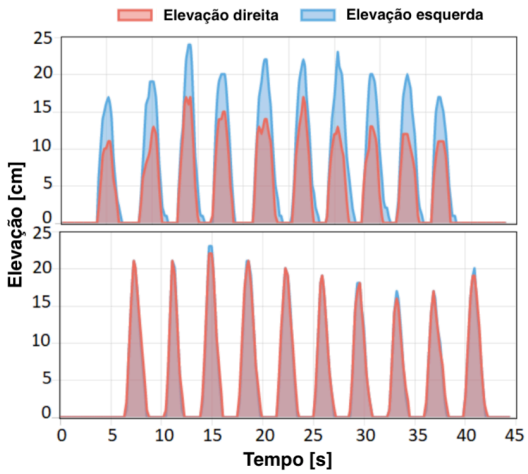


Figura 7.5 – Elevação lateral anómala (cima); elevação normal do andarilho (baixo).

7.2. Segundo ensaio

Para o segundo ensaio foi requisitado um conjunto de 3 voluntários. Sendo eles saudáveis, do do sexo masculino, com idades compreendidas entre os 20-35 anos, pesos entre os 70-90 kg, e alturas entre 1,65-1.90 metros. No que diz respeito à execução do ensaio, dois dos voluntários realizaram-no simulando ser pacientes com limitações de movimentos, e o outro realizou-o de forma normal. As limitações simuladas foram a afectação e comprometimento do lado direito do corpo (1 voluntário), e do lado esquerdo (1 voluntário) devido a AVCs. Em teoria, a manifestação dessas incapacidades poderia apresentar uma elevação mais acentuada do andarilho no lado oposto dessas limitações, devido a falta de força nos braços, e a um equilíbrio desviado para o lado dessas mesmas incapacidades e a uma limitação da rotação do torso para alterar a direção.

No que diz respeito ao plano e execução do ensaio foram definidos e realizados dois tipos de trajetos executados pelos voluntários. O primeiro, apresentado no topo da Figura 7.6, que consistiu em andar cinco metros em linha recta. E o segundo, apresentado na base da mesma figura, que teve como objetivo andar cinco metros em ziguezague. De referir ainda que neste ensaio, foram adquiridos e analisados os dados de equilíbrio do voluntário, bem como a elevação e ângulos de orientação do andarilho por si executados.

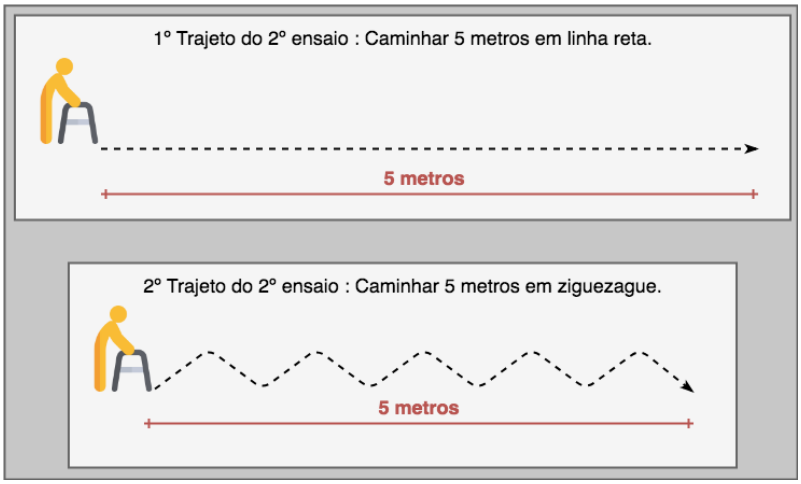


Figura 7.6 – Trajetos do 2º ensaio.

Por forma a facilitar a simplificar a leitura e compreensão, foram definidos os seguintes nomes de código para cada um dos voluntários:

- **Voluntário 1:** Representa o voluntário que simulou ter o lado esquerdo do seu corpo afectado devido a um AVC.

- **Voluntário 2:** Representa o voluntário que simulou ter o lado direito do seu corpo afectado devido a um AVC.
- **Voluntário 3:** Representa o voluntário que realizou o ensaio normalmente.

7.2.1. Resultados de equilíbrio

Em relação aos resultados de equilibrio, as Figuras 7.7 e 7.8 apresentam as performances dos três voluntários em cada um dos dois trajetos realizados. Sendo a primeira, a Figura 7.7, relativa ao primeiro trajeto, e a segunda, a Figura 7.8, ao segundo. Nos gráficos presentes nestas figuras, pode ver-se, novamente, a representação da estrutura do andarilho, vista da cima a azul, assim como conjuntos de pontos vermelhos representando os pontos de equilíbrio instantâneos obtidos pela força aplicada sobre o andarilho.

Na Figura 7.7 pode verificar-se que durante a execução do 1º trajeto (caminhada de 5 metros em linha reta), os voluntários 1 e 2 tiveram o seu equilíbrio bastante deviado do centro. Analisando com mais minúncia, verifica-se que esse desvio é realizado para o lado da sua incapacidade. Por contraposição, constata-se também que o voluntário 3, sem qualquer limitação, apresentou um conjunto de pontos de equilíbrio bastante centrado, ainda que com alguma dispersão.

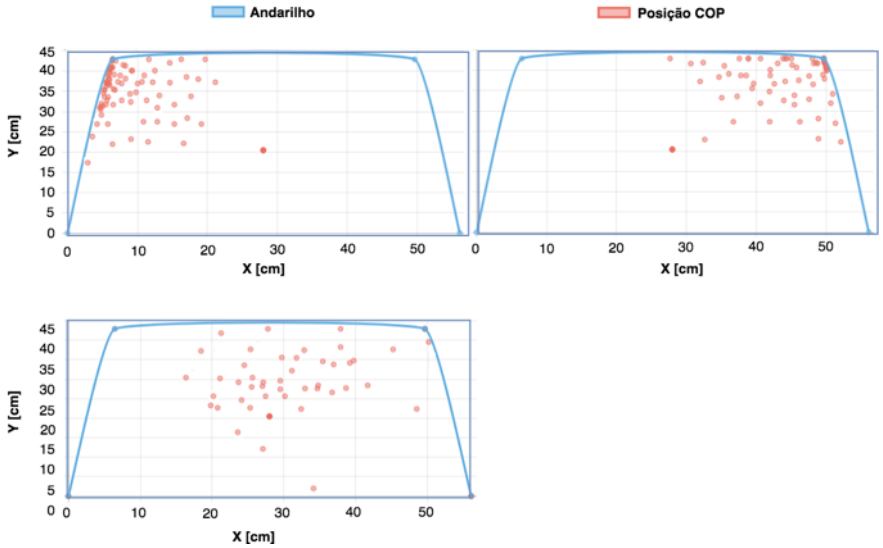


Figura 7.7 – Performance do centro de pressão no 1º exercício.

A Figura 7.8 apresenta os resultados da performance dos voluntários na execução do 2º trajeto. Nesta figura, podem encontrar-se padrões de equilíbrio semelhantes aos do 1º trajeto. No entanto, realizando uma análise mais meticulosa, verifica-se que os pontos de equilíbrio de cada voluntário estão mais concentrados neste segundo trajeto. Nesse sentido, verifica-se que os voluntários com limitações, voluntários 1 e 2, tiveram uma

performance mais limitada. Podendo esse facto ser atribuído à maior complexidade deste segundo trajecto, tendo de caminhar e mudar de direção ao mesmo tempo.

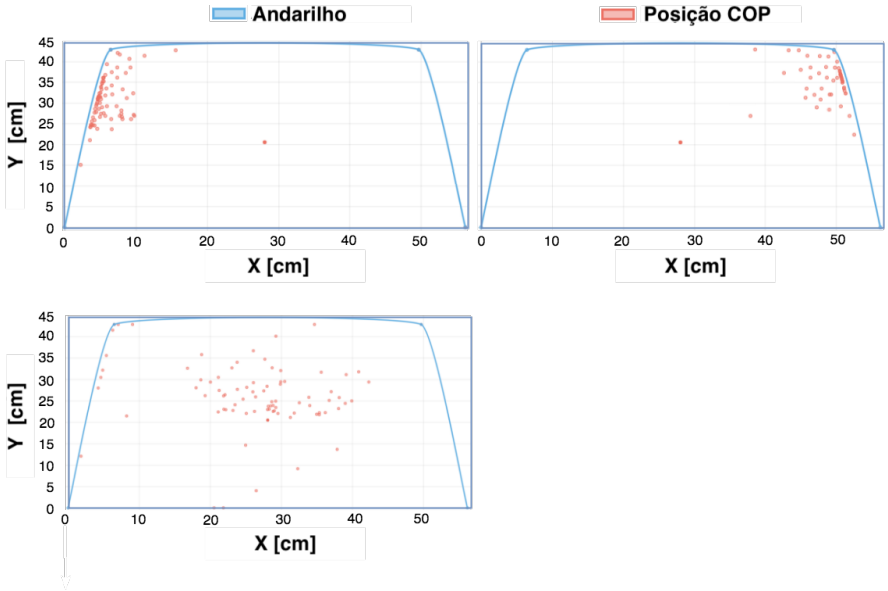


Figura 7.8 – Performance do centro de pressão no 2º exercício.

7.2.2. Resultados de elevação

A performance dos três voluntários, no que diz respeito à elevação do andarilho, é apresentada nas Figuras 7.9 e 7.10. Nestas figuras são mostrados os dados relativos ao primeiro e segundo trajetos realizados, respectivamente. A vermelho e a azul exibem-se, respectivamente, as elevações lateral direita e esquerda do andarilho. A elevação é medida em centímetros e o tempo decorrido, em segundos.

Na Figura 7.9 apresentam-se três gráficos relativos à elevação do andarilho por parte dos voluntários com incapacidades, os voluntários 1 e 2, e voluntário sem qualquer incapacidade, o voluntário 3. No que à análise diz respeito, é possível ver que o voluntário 3 realizou o trajeto com uma variação de elevação residual (em média inferior a 0.5 cm). Pode constatar-se também que os voluntários 1 e 2, manifestaram uma maior elevação do andarilho do lado oposto ao da sua limitação (superior, em média, a 3 cm). Podendo este facto ser associado a falta de força no lado da sua limitação, favorecendo o lado oposto.

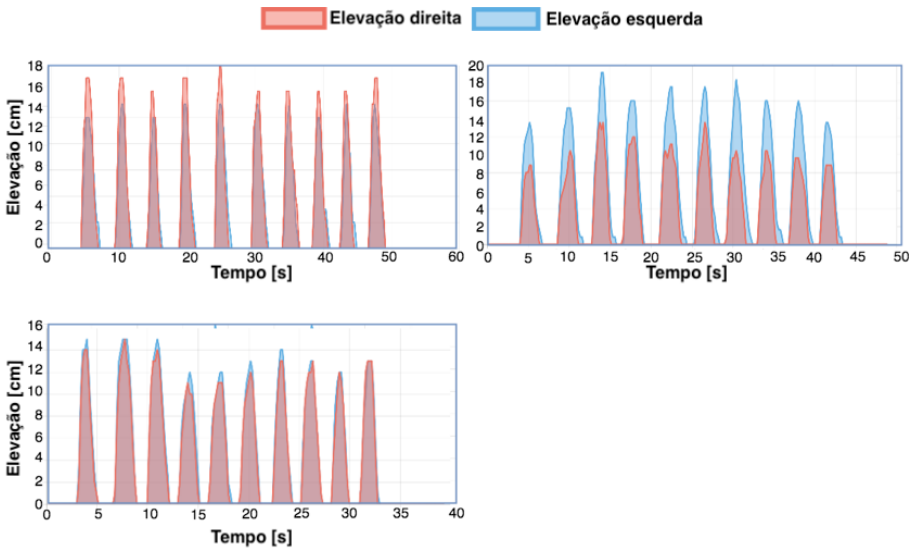


Figura 7.9 – Performance de elevação no 1º exercício.

A Figura 7.10 apresenta o mesmo padrão da Figura 7.9. Isto é, por um lado uma maior elevação do andarilho no lado oposto ao das suas limitações, no caso dos voluntários 1 e 2. E por outro lado, uma baixa variação entre as elevações do lado esquerdo e direito do andarilho, no caso do voluntário 3. No entanto, e analisando melhor, é possível verificar que neste segundo trajeto, os voluntários 1 e 2, registaram uma maior variação das elevações laterais (5 cm, em média). Esta alteração poderá dever-se à maior complexidade deste segundo trajeto, uma vez que os voluntários tiveram de rodar o torso à medida que realizavam a caminhada em ziguezague.

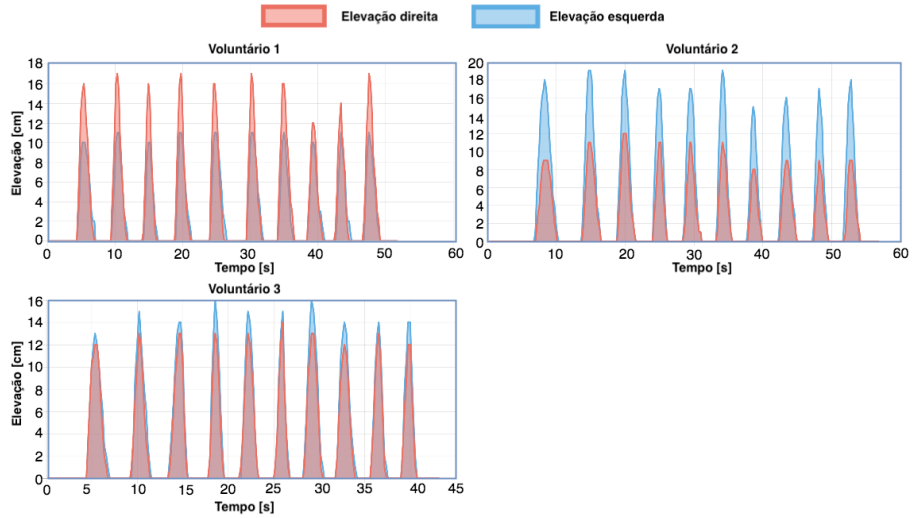


Figura 7.10 – Performance de elevação no 2º exercício.

7.2.3. Resultados de orientação

As Figuras 7.11 e 7.12 apresentam a performance dos três voluntários no que diz respeito a sua orientação na utilização do andarilho, neste segundo ensaio. Estas figuras mostram os ângulos de orientação no primeiro (ver Figura 7.11) e segundo (ver Figura 7.12) trajectos realizados. A vermelho, azul e verde exibem-se, respectivamente, os

ângulos de orientação *Roll*, *Pitch* e *Yaw* realizados pelos voluntários. Ângulos esses que são medidos em graus, e o tempo decorrido em segundos.

No 1º trajeto, com os resultados apresentados na Figura 7.11, pode constatar-se que os voluntários que reproduziram incapacidades, os voluntários 1 e 2, tiveram dificuldades em caminhar os cinco metros mantendo uma linha recta. Estes voluntários oscilaram entre as direções esquerda e direita, como se pode ver nas curvas do ângulo *Yaw* dos gráficos da Figura 7.11. Em relação ao voluntário sem limitações físicas, o voluntário 3, podem apenas ser vistas pequenas oscilações associadas ao movimento natural do corpo.

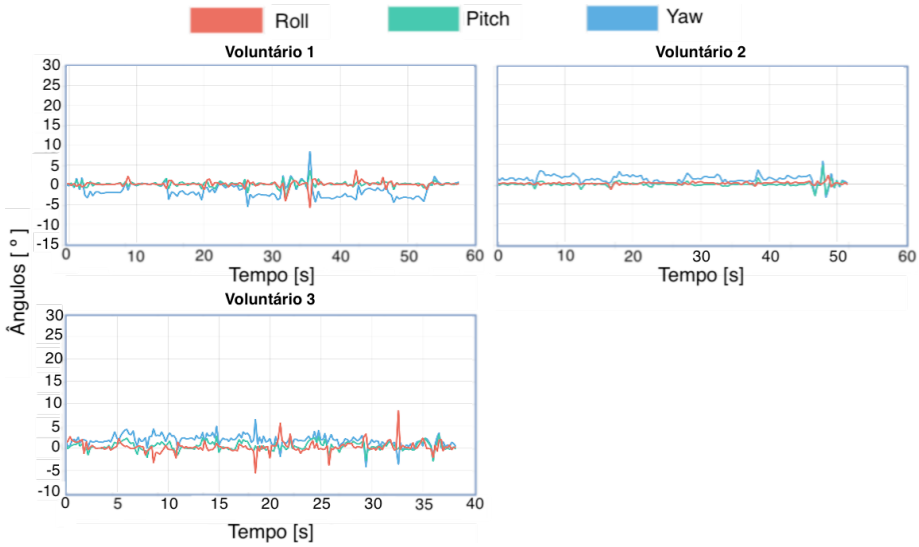


Figura 7.11 – Performance de orientação no 1º exercício.

Os resultados relativos ao segundo trajeto (Figura 7.12), revelam que os voluntários 1 e 2 manifestaram dificuldade em rodar para o lado oposto ao da sua limitação. Analisando as curvas do ângulo *Yaw* dos gráficos desses mesmos voluntários, pode ser verificado que estes realizaram uma menor rotação de torso na direção do seu lado com limitações. Sendo que este facto pode estar associado a movimentos inconscientes de forma a prevenir possíveis quedas. Por outro lado o gráfico relativo ao voluntário 3, revelou que este realizou rotações de torso (ângulo *Yaw*) semelhantes para os dois lados, esquerdo e direito.

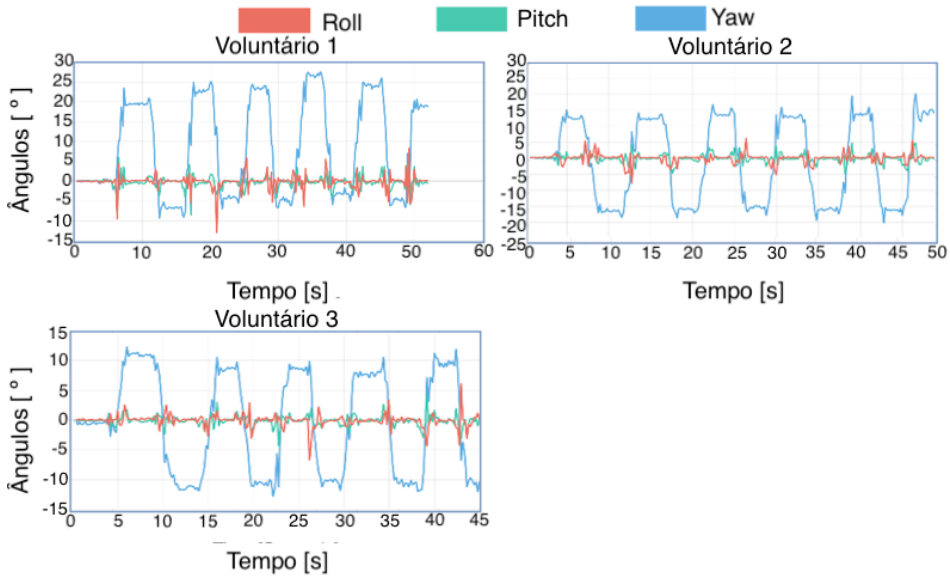


Figura 7.12 – Performance de orientação no 2º exercício.

Capítulo 8 – Conclusões e trabalho futuro

Neste capítulo pretende-se apresentar as conclusões e o trabalho futuro do projeto desenvolvido e previamente apresentado.

8.1. Conclusões

A presente dissertação apresenta o desenvolvimento de um sistema *lowcost* de reabilitação física de pacientes que tenham limitações motoras, especialmente relacionadas com a marcha e equilíbrio e derivadas de acidentes, de doenças ou da idade. Sendo o seu principal objetivo o de auxiliar os fisioterapeutas na análise da evolução dos seus pacientes de maneira a obter melhores resultados e reduzir o período de tratamento. O sistema conta com um protótipo baseado num andarilho inteligente com sensores, que adquire processa e envia métricas, relativas à marcha dos pacientes, para o servidor onde ficam disponíveis para serem acedidas e analisadas pelos utilizadores das aplicações do mesmo.

Durante o desenvolvimento do sistema foram analisados os equipamentos de fisioterapia, os sensores, os algoritmos e os tipos de aplicações mais indicados para ajudar no processo de reabilitação de pacientes. Primeiramente, e em relação aos equipamentos de fisioterapia, foi selecionado o andarilho devido a ser um dos mais utilizados na área da reabilitação da marcha. Isto deve-se ao facto de permitir manter o equilíbrio e a posição, transmitindo confiança aos pacientes, para além da existência de estudos, referidos na revisão da literatura, que indicam que estes ajudam a reduzir o tempo de reabilitação. Posteriormente, foram identificadas as métricas mais relevantes a serem adquiridas. Sendo essas a orientação do andarilho e o equilíbrio do paciente, a elevação bilateral do andarilho escolhido e o número de passos dados, uma vez que permitem a deteção de falta de mobilidade lateral, de risco de quedas e falta de força nos membros superiores. De seguida foram analisados os sensores a utilizar tendo em conta os dados físicos a adquirir e processar para a obtenção das métricas requeridas, a sua utilização noutros sistemas, e o seu custo. Nesse sentido foram escolhidos um *IMU* para a orientação, dois sensores de ultrassons para a elevação e para o cálculo do número de passos, e quatro células de carga para a deteção do equilíbrio. Finalmente, considerando os dados recolhidos, foram analisados os métodos de visualização mais adequados. Sendo que devido à natureza da profissão dos fisioterapeutas, que requer bastante mobilidade para acompanhar os seus pacientes, e ao facto de se pretender criar

um sistema abrangente, foram desenvolvidos uma aplicação móvel *Android*, a *PhysioApp* e de uma aplicação *web*, a *PhysioWebapp*. Aplicações essas que realizam pedidos *HTTP,* acedendo a ficheiros *PHP* presentes no servidor do sistema, para visualizar ou criar planos de exercícios, analisar dados das sessões de fisioterapia ou visualizar as métricas adquiridas das sessões que estejam em curso.

O sistema desenvolvido é relevante e importante uma vez que permite tanto ao fisioterapeuta, como ao paciente terem uma maior consciência da evolução do processo de reabilitação motora. O sistema possibilita a criação de tratamentos mais eficientes, obtendo melhores resultados e reduzindo o tempo de recuperação. Para além disso, o sistema disponibiliza a análise de sessões em curso, o que permite ao fisioterapeuta a correção do movimento do paciente durante a execução dos exercícios. Em relação aos ensaios realizados, estes provam as reais potencialidades do sistema, já que permitem verificar a deteção da falta de equilíbrio e de força nos braços, e do alcance de movimentos do torso do paciente. Deteções essas que possibilitam um melhor diagnóstico e a criação de planos de exercícios mais eficientes que auxiliam na melhora de qualidade de vida e independência do paciente ao fornecer-lhe um maior alcance de movimentos, um acréscimo de equilíbrio e de força, reduzindo com isso as possibilidades de quedas, que podem levar a novas maleitas.

8.2. Trabalho futuro

O sistema desenvolvido e mostrado ao longo da presente dissertação apresenta-se como viável e benéfico tanto para os pacientes como para os fisioterapeutas. Esta solução é conveniente no acompanhamento dos treinos de fisioterapia da marcha e equilíbrio, assim como no apoio às análises das performances dos doentes, e consequentes elaborações de planos de exercícios, por parte dos fisioterapeutas. No entanto, e apesar deste ter sido um trabalho moroso e ambicioso devido à sua complexidade, existem aspectos que ainda podem ser melhorados assim como novas funções e/ou características que podem ser adicionadas. Algumas dessas características são apresentadas abaixo:

- Inclusão de um sensor leitor de impressões digitais de maneira ao paciente poder realizar as sessões de fisioterapia, ainda que não tenha consigo o seu cartão do sistema.

- Inclusão de sensores *ECG* (Electrocardiograma) de maneira a monitorizar as condições cardiorrespiratórias durante as sessões de fisioterapia.
- Alteração do sistema de maneira a torná-lo mais abrangente, podendo aplicá-lo à monitorização da atividade diária no âmbito da *Ambient Assisted Living*.
- Utilização do protocolo *MQTT* como protocolo de comunicação do sistema.
- Utilização de ferramentas de *Big Data* aplicadas à *IoT* e à reabilitação física de maneira a desenvolver modelos de predição da evolução dos pacientes durante as sessões de fisioterapia baseados nos dados recolhidos.
- Utilização de algoritmos de *Data Mining* para identificar e classificar padrões de marcha, de predominância de elevação lateral ou de desvio do equilíbrio durante uma sessão ou um tratamento completo de fisioterapia.

Bibliografia

[1] L. Vogt, K. Lucki, M. Bach, and W. Banzer, "Rollator use and functional outcome of geriatric rehabilitation," *J. Rehabil. Res. Dev.*, vol. 47, no. 2, p. 151, 2010.

[2] O. Postolache, "Physical rehabilitation assessment based on smart training equipment and mobile APPs," *2015 E-Health Bioeng. Conf. EHB 2015*, 2016.

[3] C. A. Cifuentes, C. Rodriguez, A. Frizera-Neto, T. F. Bastos-Filho, and R. Carelli, "Multimodal Human-Robot Interaction for Walker-Assisted Gait," *IEEE Syst. J.*, pp. 1–11, 2014.

[4] R. Gerena *et al.*, "Wireless Instrumented Walker for Remote Rehabilitation Monitoring," 2015.

[5] S. Yoshida, "A Global Report on Falls Prevention Epidemiology of Falls," *WHO Rep.*, pp. 1–40, 2007.

[6] G. Häggblom-Kronlöf and U. Sonn, "Use of assistive devices - A reality full of contradictions in elderly persons' everyday life," *Disabil. Rehabil. Assist. Technol.*, vol. 2, no. 6, pp. 335–345, 2007.

[7] ONU, "World population, ageing," *Suggest. Cit. United Nations, Dep. Econ. Soc. Aff. Popul. Div. (2015). World Popul. Ageing*, vol. United Nat, no. (ST/ESA/SER.A/390, p. 164, 2015.

[8] World Health Organization, "WHO Global Report on Falls Prevention in Older Age.," *Community Health (Bristol).*, p. 53, 2007.

[9] S. Page, K. R. Mun, Z. Guo, F. A. Reyes, H. Yu, and V. Pasqui, "Unbalance detection to avoid falls with the use of a smart walker," *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics*, vol. 2016–July, pp. 1000–1005, 2016.

[10] S. Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24(3), no. 3, pp. 45–78, 2007.

[11] M. States, "Rehabilitation 2030 : A Call for Action Concept Note," pp. 2016–2017, 2016.

[12] T. Alkjær, P. K. Larsen, G. Pedersen, L. H. Nielsen, and E. B. Simonsen, "Biomechanical analysis of rollator walking," *Biomed. Eng. Online*, vol. 5, pp. 1–7, 2006.

[13] WHO, "World Report on Disability - Summary," *World Rep. Disabil. 2011*, no. WHO/NMH/VIP/11.01, pp. 1–23, 2011.

[14] T. Bardsley, G., Blocka, D., Borg, J., Brintnell, S., Constantine, D., Dillu, A.,… Verhoeff, "Joint position paper on the provision of mobility devices JOINT POSITION PAPER ON THE PROVISION OF MOBILITY DEVICES IN LESS-RESOURCED SETTINGS," *World Heal. Organ.*, vol. 5, no. 2, 2011.

[15] World Health Organization, "Relatório mundial sobre a deficiência," *Organ. Mund. da Saúde*, p. 334, 2012.

[16] L. De Vito, O. Postolache, and S. Rapuano, "Measurements and sensors for motion tracking in motor rehabilitation," *IEEE Instrum. Meas. Mag.*, vol. 17, no. 3, pp. 30–38, 2014.

[17] R. C. A. Alves, L. B. Gabriel, B. T. De Oliveira, C. B. Margi, and F. C. L. Dos Santos, "Assisting physical (Hydro) therapy with wireless sensors networks," *IEEE Internet Things J.*, vol. 2, no. 2, pp. 113–120, 2015.

[18] T. S. Jesus and I. L. Silva, "Toward an evidence-based patient-provider communication in rehabilitation: Linking communication elements to better rehabilitation outcomes.," *Clin. Rehabil.*, p. 0269215515585133-, 2015.

[19] D. Capecci, S. H. Kim, K. B. Reed, and I. Handzic, "Crutch tip for swing-through crutch walking control based on a kinetic shape," *IEEE Int. Conf. Rehabil. Robot.*, vol. 2015–Septe, pp. 612–617, 2015.

[20] O. Postolache and P. S. Girao, "Physiotherapy smart connected devices for S-health," *IISA 2016 - 7th Int. Conf. Information, Intell. Syst. Appl.*, 2016.

[21] C. Y. Su, C. Y. Hsiao, R. G. Lee, and J. H. Chen, "TaoBall: An interactive IoT ball design for rehabilitation," *IEEE Int. Conf. Consum. Electron. - Berlin, ICCE-Berlin*, vol. 2016–Octob, pp. 24–27, 2016.

[22] J. M. D. Pereira, O. Postolache, V. Viegas, and P. S. Girao, "A low cost measurement system to extract kinematic parameters from walker devices," *Conf. Rec. - IEEE Instrum. Meas. Technol. Conf.*, vol. 2015–July, pp. 1991–1996, 2015.

[23] R. De Souza, V. C. Cavalcanti Roza, and O. Postolache, "A Multi-sensing Physical Therapy Assessment for Children with Cerebral Palsy," *2017 Elev. Int. Conf. Sens. Technol.*, pp. 1–6, 2017.

[24] O. Postolache, P. S. Girao, J. M. D. Pereira, and G. Postolache, "Wearable system for gait assessment during physical rehabilitation process," *2015 9th Int.*

Symp. Adv. Top. Electr. Eng., pp. 321–326, 2015.

[25] D. Ferreira, R. Oliveira, and O. Postolache, "Physical Rehabilitation based on Kinect Serious," pp. 0–5, 2017.

[26] J. W. Qiu *et al.*, "Continuous human location and posture tracking by multiple depth sensors," *Proc. - 2014 IEEE Int. Conf. Internet Things, iThings 2014, 2014 IEEE Int. Conf. Green Comput. Commun. GreenCom 2014 2014 IEEE Int. Conf. Cyber-Physical-Social Comput. CPS 20*, no. iThings, pp. 155–160, 2014.

[27] R. N. Madeira, L. Costa, and O. Postolache, "PhysioMate-Pervasive physical rehabilitation based on NUI and gamification," *EPE 2014 - Proc. 2014 Int. Conf. Expo. Electr. Power Eng.*, no. Epe, pp. 612–616, 2014.

[28] N. Duarte, O. Postolache, and J. Sharcanski, "KSGphysio – Kinect Serious Game for Physiotherapy," *Int. Conf. Expo. Electr. Power Eng.*, no. Epe, pp. 16–18, 2014.

[29] M. Martins, C. P. Santos, S. Page, L. Saint-Bauzel, V. Pasqui, and A. Meziere, "Real-Time gait assessment with an active depth sensor placed in a walker," *IEEE Int. Conf. Rehabil. Robot.*, vol. 2015–Septe, pp. 690–695, 2015.

[30] C. A. Cifuentes, C. Rodriguez, A. Frizera-Neto, T. F. Bastos-Filho, and R. Carelli, "Multimodal Human-Robot Interaction for Walker-Assisted Gait," *IEEE Syst. J.*, vol. 10, no. 3, pp. 933–943, 2016.

[31] A. Wachaja, P. Agarwal, M. Zink, M. R. Adame, K. Möller, and W. Burgard, "Navigating blind people with walking impairments using a smart walker," *Auton. Robots*, vol. 41, no. 3, pp. 555–573, 2017.

[32] I. Bojanova, G. Hurlburt, and J. Voas, "Imagineering an internet of anything," *Computer (Long. Beach. Calif).*, vol. 47, no. 6, pp. 72–77, 2014.

[33] J. Kietzmann and K. Robson, "Introduction to the internet of everything: Connecting people, things, and data Minitrack," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2016–March, p. 3918, 2016.

[34] B. Dorsemaine, J. P. Gaulier, J. P. Wary, N. Kheir, and P. Urien, "Internet of Things: A Definition and Taxonomy," *Proc. - NGMAST 2015 9th Int. Conf. Next Gener. Mob. Appl. Serv. Technol.*, pp. 72–77, 2016.

[35] D. V. Dimitrov, "Medical internet of things and big data in healthcare," *Healthc. Inform. Res.*, vol. 22, no. 3, pp. 156–163, 2016.

[36] R. Jesner Clarke, "Smart Cities and the Internet of Everything: The Foundation for Delivering Next-Generation Citizen Services," *Alexandria, VA, Tech. Rep*,

no. October, pp. 1–18, 2013.

[37] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT)," *2015 Internet Technol. Appl. ITA 2015 - Proc. 6th Int. Conf.*, pp. 219–224, 2015.

[38] K. Ashton, "That 'Internet of Things' Thing," *RFiD J.*, p. 4986, 2009.

[39] ITU, "The Internet of Things," *Itu Internet Rep. 2005*, p. 212, 2005.

[40] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, 2014.

[41] K. Gusmeroli, S., Haller, S., Harrison, M., Kalaboukas, K., Tomasella, M., Vermesan, O., & Wouters, *Vision and challenges for realizing the internet of things*, vol. 1, no. APRIL. 2009.

[42] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," *Proc. - 10th Int. Conf. Front. Inf. Technol. FIT 2012*, pp. 257–260, 2012.

[43] S. M. R. Islam, D. Kwak, H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care : A Comprehensive Survey," *Access, IEEE*, vol. 3, pp. 678–708, 2015.

[44] Statista, "• Mobile OS market share 2018 | Statista." [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. [Accessed: 12-Sep-2018].

[45] Android, "Android, the world's most popular mobile platform | Android Developers," 2018. [Online]. Available: https://developer.android.com/about/index.html. [Accessed: 16-Feb-2018].

[46] Diffen, "Android vs iOS - Difference and Comparison | Diffen." [Online]. Available: https://www.diffen.com/difference/Android_vs_iOS. [Accessed: 12-Sep-2018].

[47] N. Tiwari, "How open sourcing Android made it a mobile market leader | Opensource.com." [Online]. Available: https://opensource.com/business/14/7/how-open-sourcing-android-made-it-mobile-market-leader. [Accessed: 12-Sep-2018].

[48] Statista, "App stores: number of apps in leading app stores 2018 | Statista." [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed: 12-Sep-2018].

[49] C. L. Ventola, "Mobile Devices and Apps for Health Care Professionals: Uses

and Benefits.," *Pharm. Ther.*, vol. 39, no. 5, pp. 356–364, 2014.

[50] P. Leite and O. Postolache, "Gait rehabilitation monitor," *2017 E-Health Bioeng. Conf. EHB 2017*, pp. 438–441, 2017.

[51] F. Lourenço, O. Postolache, and G. Postolache, "Tailored virtual reality and mobile application for motor rehabilitation," *2018 IEEE Int. Instrum. Meas. Technol. Conf.*, pp. 1–6, 2018.

[52] Arduino - ArduinoBoardMega, "Arduino Mega 2560 Rev3," 2018. [Online]. Available: https://store.arduino.cc/arduino-mega-2560-rev3. [Accessed: 16-Feb-2018].

[53] Pololu, "Pololu - MinIMU-9 v3 Gyro, Accelerometer, and Compass (L3GD20H and LSM303D Carrier)," 2018. [Online]. Available: https://www.pololu.com/product/2468. [Accessed: 16-Feb-2018].

[54] "Ultrasonic Ranging Module HC-SR04."

[55] Sparkfun, "Ultrasonic Sensor - HC-SR04 - SEN-13959 - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/products/13959. [Accessed: 29-Aug-2018].

[56] RobotShop, "50 Kg Micro Load Cell - RobotShop," 2017. [Online]. Available: https://www.robotshop.com/en/micro-load-cell-50-kg.html. [Accessed: 16-Feb-2018].

[57] NXP Semiconductors, "MFRC522 Standard performance MIFARE and NTAG frontend," no. 3.9, p. 95, 2016.

[58] R. Solutions, "HC-05 Bluetooth Module User 's Manual V1 . 0," 2015.

[59] Sparkfun, "WiFi Module - ESP8266 - WRL-13678 - SparkFun Electronics," 2018. [Online]. Available: https://www.sparkfun.com/products/13678. [Accessed: 19-Feb-2018].

[60] Pololu, "l3g-arduino." [Online]. Available: https://github.com/pololu/l3g-arduino. [Accessed: 29-Aug-2018].

[61] Pololu, "lsm303-arduino." [Online]. Available: https://github.com/pololu/lsm303-arduino. [Accessed: 29-Aug-2018].

[62] Martin Baker, "Maths - Euler Angles." [Online]. Available: http://www.euclideanspace.com/maths/geometry/rotations/euler/index.htm. [Accessed: 29-Aug-2018].

[63] CH Robotics, "Understanding Euler Angles | CH Robotics." [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. [Accessed: 29-

Aug-2018].

[64] L. Tran and S. L. Obispo, “Data Fusion with 9 Degrees of Freedom Inertial Measurement Unit To Determine Object’s Orientation,” 2017.

[65] H. Choi, “Henry Choi: Coarse roll and pitch initialization using accelerometer in Unity,” 2017. [Online]. Available: http://henryomd.blogspot.com/2017/02/coarse-roll-and-pitch-initialization.html?m=1. [Accessed: 28-Sep-2018].

[66] TKJElectronics, “KalmanFilter.” [Online]. Available: https://github.com/TKJElectronics/KalmanFilter. [Accessed: 29-Aug-2018].

[67] jsvester, “Simple Arduino and HC-SR04 Example: 3 Steps.” [Online]. Available: https://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/. [Accessed: 29-Aug-2018].

[68] IBM, “Introduction to LAMP technology,” 2005. [Online]. Available: https://www.ibm.com/developerworks/web/tutorials/wa-lamp/wa-lamp.html. [Accessed: 16-Feb-2018].

[69] J. Wallen, “Easy LAMP Server Installation | Linux.com | The source for Linux information,” 2010. [Online]. Available: https://www.linux.com/learn/easy-lamp-server-installation. [Accessed: 16-Feb-2018].

[70] MySQL, “MySQL :: MySQL Workbench,” 2018. [Online]. Available: https://www.mysql.com/products/workbench/. [Accessed: 16-Feb-2018].

[71] “phpMyAdmin.” [Online]. Available: https://www.phpmyadmin.net/. [Accessed: 28-May-2018].

[72] “Introducing JSON.” [Online]. Available: https://www.json.org/. [Accessed: 13-Sep-2018].

[73] Latitude Technolabs, “Native Mobile Apps vs Hybrid Mobile Apps (Pros and Cons) - Latitude Technolabs.” [Online]. Available: https://latitudetechnolabs.com/native-mobile-apps-vs-hybrid-mobile-apps-pros-cons/. [Accessed: 17-Sep-2018].

[74] A. MacQuarrie, “Hybrid Apps vs. Native Apps: Which Should You Build? | The Manifest.” [Online]. Available: https://themanifest.com/app-development/hybrid-apps-vs-native-apps-which-should-you-build. [Accessed: 17-Sep-2018].

[75] International Data Corporation, “IDC: Smartphone OS Market Share,” 2018. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os. [Accessed: 16-Feb-2018].

[76] Mdbootstrap, “bootstrap-material-design.” [Online]. Available: https://github.com/mdbootstrap/bootstrap-material-design. [Accessed: 29-Aug-2018].

[77] SQLite, “About SQLite.” [Online]. Available: https://www.sqlite.org/about.html. [Accessed: 29-Aug-2018].

[78] SQLite, “SQLite Copyright.” [Online]. Available: https://www.sqlite.org/copyright.html. [Accessed: 29-Aug-2018].

[79] SQLite, “Appropriate Uses For SQLite.” [Online]. Available: https://www.sqlite.org/whentouse.html. [Accessed: 29-Aug-2018].

[80] SQLite, “Datatypes In SQLite Version 3.” [Online]. Available: https://www.sqlite.org/datatype3.html. [Accessed: 29-Aug-2018].

[81] Google, “Volley.” [Online]. Available: https://github.com/google/volley. [Accessed: 29-Aug-2018].

[82] PhilJay, “MPAndroidChart.” [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed: 29-Aug-2018].

[83] Chartjs, “Chart.js.” [Online]. Available: https://github.com/chartjs/Chart.js. [Accessed: 29-Aug-2018].

Apêndices

Apêndice A – Artigos

1º artigo: *Smart Walker based IoT Physical Rehabilitation System*

Este artigo foi aceite e apresentado na 1ª edição do *ISSI* (*International Symposium in Sensing and Instrumentation in IoT Era*), uma conferência organizada pela *Shangai Maritime University* em cooperação com o *IEEE Instrumentation Measurement Society*, nos dias 6 e 7 de setembro de 2018, na *Shangai Maritime University*, em Shangai, na China.

ISSI 2018 1st International Symposium on Sensing and Instrumentation in IoT Era

September 6-7, 2018 - Grand Trustel Purple Mountain Hotel, Shanghai, China

ORGANIZERS

Honorary Chairs

Youfang Huang
Shanghai Maritime University, China

Wei Yan
Shanghai Maritime University, China

General Chairs

Octavian Postolache
ISCTE-IUL & IT, Portugal

Yongsheng Yang
Shanghai Maritime University, China

Xin Wang
Fudan University, China

Technical Program Chairs

Subhas Mukhopadhyay
Macquarie University, Australia

Domenico Capriglione
Università degli Studi di Salerno, Italy

Daqi Zhu
Shanghai Maritime University, China

Publication Chairs

Bin Yang
Shanghai Maritime University, China

Huafeng Wu
Shanghai Maritime University, China

Daofang Chang
Shanghai Maritime University, China

Local Organization Chairs

Chaofeng Li
Shanghai Maritime University, China

Ziyu Lu
Shanghai Pudong Federation of R&D Institutions, China

IMPORTANT DATES

May 11, 2018
Special Session Workshop & Proposals Deadline

June 1, 2018
Paper Submission Deadline

June 25, 2018
Paper Acceptance Notification Deadline

July 15, 2018
Camera-ready Submission & Early Registration Deadline

Call for Papers

ISSI 2018 represents a forum where researchers, scientists, engineers and practitioners around the world will promote the advances in sensing and instrumentation for IoT. The symposium valorizes new and original research in IoT applications fields, such as smart city, transportation, healthcare and farming.

The full version papers (4 to 6 pages) submitted to ISSI 2018 will be accepted based on peer review process. The review process may conduct to some revision requests for the final version of the paper that must be mandatory fulfilled to have the paper accepted for the symposium proceedings.

The accepted papers will be published in conference proceedings and will appear in **IEEE Xplore** and **EI Compendex**. Selected papers will be considered for publication in special issue of **Sensors**, SCI open Journal.

Topics of the symposium will include, but are not limited to:

- Smart Sensors and Wireless Sensor Networks for IoT;
- Test and Automated Instrumentation for IoT;
- Localization, Algorithms for Smart Sensor Network;
- Software Platforms and Middleware in Smart IoT Systems;
- IoT in Retail Logistics;
- IoT in Supply Chain Finance and Management;
- Computer Vision and Application in IoT;
- IoT & Wearable Solutions for Healthcare;
- IoT System Design Methodologies;
- Big Data Processing in IoT Systems;
- Energy Harvesting and Scavenging for Wireless Sensors Networks and IoT;
- Education and Training for Networked Instrumentation and Measurement;
- Standards for IoT and IoT Security;
- Sensors and IoT for Smart Ports and Logistics;
- Industrial Internet of Things;
- Sensors and IoT for Structural and Infrastructural Health Monitoring;
- IoT for Healthcare;
- IoT Applications in Smart Cities;
- IoT Applications for Smart Home;
- IoT in Smart Farming;

Many other new initiatives and opportunities to encourage your active participation in the conference are planned, that will make ISSI 2018 a vibrant event to meet with people in sensing and instrumentation field.

Visit the conference website for each specific call and additional news:

http://issi2018.csp.escience.cn

IEEE
IEEE INSTRUMENTATION & MEASUREMENT SOCIETY

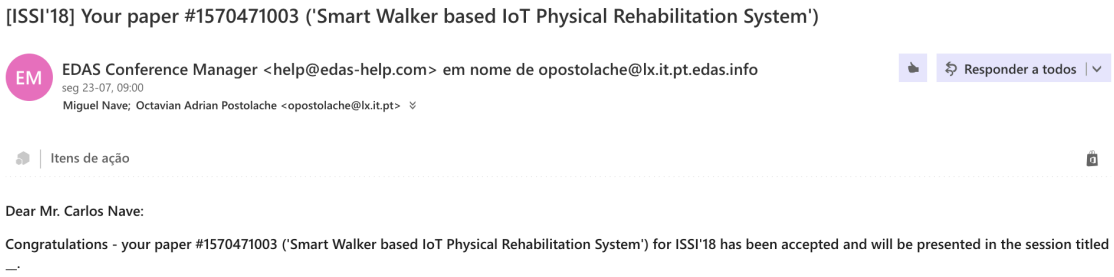
上海海事大学 Shanghai Maritime University

上海市计算机学会 Shanghai Computer Society

ISCTE IUL Instituto Universitário de Lisboa

instituto de telecomunicações

O e-mail de aceitação do artigo por parte da organização da conferência é apresentado abaixo:



[ISSI'18] Your paper #1570471003 ('Smart Walker based IoT Physical Rehabilitation System')

EM EDAS Conference Manager <help@edas-help.com> em nome de opostolache@lx.it.pt.edas.info

seg 23-07, 09:00

Miguel Nave; Octavian Adrian Postolache <opostolache@lx.it.pt>

Responder a todos

Itens de ação

Dear Mr. Carlos Nave:

Congratulations - your paper #1570471003 ('Smart Walker based IoT Physical Rehabilitation System') for ISSI'18 has been accepted and will be presented in the session titled —.

O programa da sessão da conferência onde o artigo foi apresentado encontra-se abaixo:

S1-6: IoT for Healthcare – II

Chair: Subhas Mukhopadhyay and Xiuwen Fu, Room: Golden Mountain Ballroom

3:50 PM *A Simple technique for heart sound detection and identification using kalman filter in real time analysis*
Joyanta Kumar Roy, Nirupama Mandal, Tanmay Sinha Roy and Octavian Adrian Postolache

4:05 PM *Smart Walker based IoT Physical Rehabilitation System*
Carlos Nave and Octavian Adrian Postolache

4:20 PM *Wearable and IoT Technologies Application for Physical Rehabilitation*
Ricardo Alexandre and Octavian Adrian Postolache

4:35 PM *Mobile Application based on Wireless Sensor Network for Physical Rehabilitation*
Pedro Frango and Octavian Adrian Postolache

4:50 PM *A Medical-IoT based Framework for eHealth Care*
Sandeep Pirbhulal, Wanqing Wu, Subhas Mukhopadhyay and Guanglin Li

36

Smart Walker based IoT Physical Rehabilitation System

Carlos Nave
DCTI
ISCTE-IUL
Lisbon,Portugal
cmara1@iscte-iul.pt

Octavian Postolache
DCTI
Instituto de Telecomunicações/ISCTE-IUL
Lisbon, Portugal
opostolache@lx.it.pt

***Abstract*—This paper describes the development of an IoT Physical Rehabilitation solution based on Smart Walkers. Multimodal sensing solution including IMU, Load Cells and Ultrasound sensor was designed and implemented. The smart walker is characterized by Arduino Mega computation platform that performs the calculation of walking metrics during a rehabilitation session and stores them in the cloud. The data analysis and data visualization is supported by an implemented website and mobile app. Experimental results performed with healthy volunteers and appropriate data analysis are included in the paper.**

Keywords—IoT, Sensors, Smart Walker, Arduino platform, Physical Rehabilitation.

I. Introduction

According to a WHO report, 15% of the world's population suffers from some kind of motor impairment that restricts their mobility. Among the causes that lead to a disability, the aging process stands out since the probability of disability increases with age. WHO also states that the physical rehabilitation while important to archive a sustainable development, is still inaccessible to a vast range of people worldwide, given his high cost. To overcome that, this organization promotes the development of technological devices to assist both patient and professionals [1-2].

The aging factor becomes worrying as it's leading us to an aged society, and since it's expected that by 2050 the world's elderly population, and with physical rehabilitation needs, will double present values [3].

This elderly population, due to the normal physical limitations with aging that leads to falls, need physical rehabilitation treatments not only to recover from impairments but also to avoid to acquire them in the first place, and to improve their independence to do their day-by-day activities [1],[4-6].

There has been research combining the healthcare and IoT fields to present rehabilitation solutions to people with impairments, including an upper limbs rehabilitation systems using ECG and motion sensors and machine learning algorithms to classify the rehabilitation level progress, and a fall detection system using a motion sensor to detect and prevent falls [7-8].

However, the elders rehabilitation sessions are usually performed using tools such as walkers and crutches, since they allow the balance and position maintenance during exercises. Furthermore, studies reveal that those tools not only do not interfere in the results but in some cases reduce the recovery time. Nevertheless, there has been also development of smart physical rehabilitation tools, such as smart walkers and crutches, since the degree impairment and the recovery progress can be hard to see to the naked eye [9-11].

There has been research and development of healthcare IoT solution to improve the quality of life of the ones in need. Three examples of this type of solutions are presented below.

A wireless smart walker system development based on microcontrollers, using motion, force and/or Doppler sensors to improve balance , get patient's orientation and displacement, as well as to extract motion patterns to evaluate the treatment effectiveness is described [10],[12].

There is a paper that describes another smart walker system based on Atmel microcontrollers, that uses load cell sensors, motion analysis and EMG signals to acquire upper limbs load bearing patterns [13].

The solution described in the current paper presents some original and/or improved features when compared with the already mentioned researches. Regarding the acquired data, the developed system proposal shows upper limb data through the acquisition of the walker bilateral elevation during sessions. Respecting the walker-server communication, the proposed system shows versatility since it uses both Wi-Fi and Bluetooth communication protocols, that allows the physiotherapy sessions to proceed even when there's no Wi-Fi signal, using a Bluetooth module and a mobile device with the system's app as a gateway to the server (3G/4G). Furthermore, the presented system also enables the physiotherapists to see session's live data as well as to create workout plans and compare the patient's last 5 sessions performance.

These type of real-time systems, the patient-physiotherapist interoperability and the real-time exercise analysis improve the physiotherapist-patient relationship, making possible to obtain better an quicker results, decreasing recovery time and consequently treatment costs [11],[14].

II. System Description

The solution described in this paper is a smart walker as part of physiotherapy system. The smart walker is developed around an Atmega2560 microcontroller platform board (Edge Layer)

978-1-5386-5638-9/18/$31.00 ©2018 IEEE

and acquires the data from the sensors connected to the microcontroller inputs (Sensor Layer), and then stores them in a cloud server database. The node can communicate directly with the cloud (with a Wi-Fi module), or through a Mobile App (Bluetooth module) installed in an Android OS device. In the last case, the system can show the live data in the mobile app (Fig. 1).

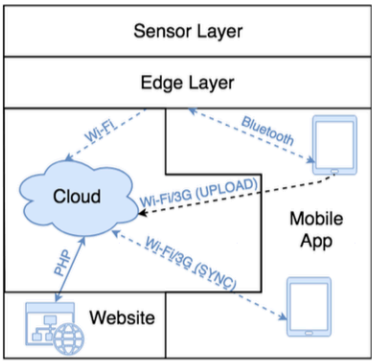


Fig. 1. System's architecture.

A. System Hardware

The smart walker node is the hardware part of the system which is composed by an Atmega2560 microcontroller (which acts as the system's edge), with a signal conditioning circuit (SCC), communication modules and analog/digital sensors for distance (D1-D2) and force (F1-F4), as can be seen in Fig. 2.

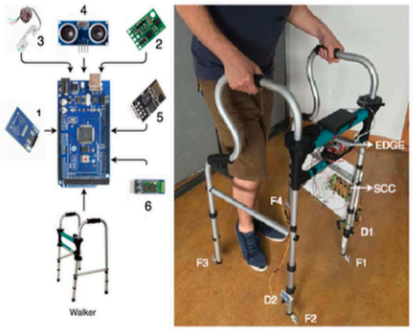


Fig. 2. Smart Walker node diagram (left) with, RFID (1), IMU (2), force sensor (3), distance sensor (4), wi-fi (5) and bluetooth (6); Smart Walker prototype (right).

1) RFID Reader

This 13.56 MHz RFID sensor has a 5 cm reading range and uses the SPI protocol to communicate with the computation board and it's used in the system to read the patient's id card and start a physiotherapy smart walker session.

2) Orientation Sensor (IMU)

The inertial measurement system IMU Pololu MinIMU-9 v3 is a 9 degrees of freedom sensor, with an accelerometer, a magnetometer and a gyroscope, all with 3 axis. This sensor was used to get the patient's orientation angles during the training session through the Euler angles (roll, pitch, and yaw).

3) Force Sensors

Four Micro Load Cells – RB-Phi-120 sensors are used to acquire the force applied by the patient in each walker leg, as well as the patient center of force associated with patient gait and balance.

4) Distance Sensor

Two HC-SR04 ultrasonic sensors are used to extract the walker's elevation in each side by the patient during a physiotherapy session. This measurement is done so the physiotherapist can notice if the patient was elevating the walker more in one side that the other, which can be associated to an arm being weaker than the other.

5) Wi-Fi communication module

The ESP-01 module is used to transmit the acquired sensor's data to the cloud server database, when it's used a card to start a session.

6) Bluetooth Communication Module

The HC-05 module sends the live data to a mobile with the system's app, to plot and store the data, when the session is initiated in the app.

B. System Software

The system has three components of software. The embedded software, the mobile software and the server software. The first one is the code that runs in the system's node, and acquires and sends all the needed metrics. The second one, the cloud server software is composed by the database, where the system's data is stored, the system's website, where the user can access to the system's data that is available to him. Finally, is the mobile application, where the registered users can see the system's *data* that is available to them.

1) Embedded software

This software is the C language code that runs in the node computing platform, the Arduino Mega, acquires the data from the sensors attached to it, and computes them to get the desired metrics. After that it sends those metrics directly, or through the mobile app, to the cloud server using the Bluetooth or Wi-Fi communication modules (Fig. 1).

2) Cloud Server Software

The cloud server stores and manages the system's data. For that it was created and stored in the cloud server, a MySQL database to store all the system's data, PHP script files to interact with da database and to insert/retrieve data to/from the database, and a website where the user can visualize all the data that he is allowed to see.

The database is a MySQL database, which has 9 tables to represent and relate the 3 types of users, the sessions and it's sensors data, and the patients workouts and it's exercises, as can be seen in Fig. 3.

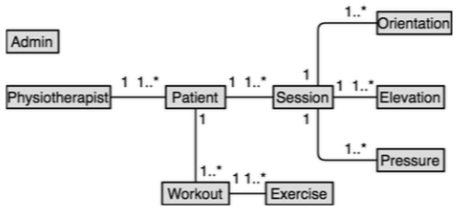


Fig. 3. System's database EDR diagram.

There are also present in the system's cloud server PHP script files to insert and retrieve data to and from the system's database. This files are accessed by the network node (ESP-01 Wi-Fi module), the website and the mobile app, through HTTP POST Request queries to insert/get the desired data. A sample of one of the used PHP script code, responsible for the session's creation, is shown in Fig. 4.

```
<?php
   $con = mysqli_connect($host, $user, $pass, $db);
   $sql = "INSERT INTO session (session_date, session_start,
           session_finish, patient_patient_id,
           patient_physiotherapist_physio_id)
           VALUES ('$date', '$time', 'NULL','$patient_id',
                   '$physio_id')";

   $result_session = $con->query($sql);

   if($result_session == true){
      $session_id = $con->insert_id;
      $result[] =  array("auth"=>true,"success"=>true,
                   "session_id"=>$session_id);
   }
   else{
      $result[] = array("auth"=>true,"success"=>false);
   }
   echo json_encode($result);
?>
```

Fig. 4. PHP script of session creation.

The cloud server also stores a website, that uses the already described PHP script files so the user can access the allowed data. This is a full responsive HTML5 website, which means that fits all devices and screen resolutions. A page of the website is shown in the left side of Fig. 5.

Fig. 5. Website (left) and mobile app (right) interface.

To access the website contents one has to be a registered user. The user has one of three user types. A physiotherapist, a patient or an administrator, each one with different data access permissions, and allowed actions (Fig. 6).

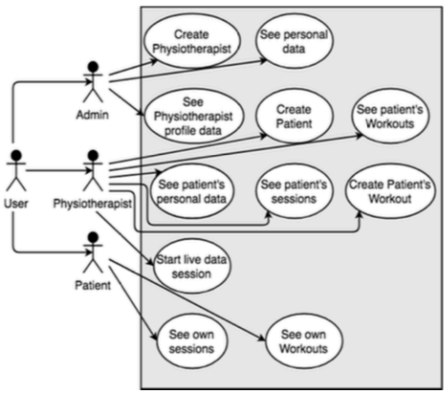


Fig. 6. System's user functionalities.

3) Mobile App

The mobile application was developed for Android devices, since it's the users choice leader for many years, registering in the first quarter of 2017 a 85% percentage of the mobile devices market share [15].

In the App the user can perform all the actions that he can do in the website, with the addition, if the user is a physiotherapist, of being able of starting a live session, plotting the patient's real time data directly in the app, and storing that data in the cloud server, as shown in the right side of Fig. 5.

III. Sensing Algorithms

In this section it's presented the calculus done in the embedded software to acquire and store the desired metrics in the cloud server.

A. Orientation Angles

To get the patient's orientation angles, or Euler angles, were used the pitch (1), roll (2) and yaw (3) equations relating the accelerometer, magnetometer and gyroscope raw data given by the used IMU sensor.

In (1), a_x stands for acceleration in the X axis, g stands for the gravitational constant, and asin to a trigonometrical function, giving P, the Pitch angle.

$$P = \text{asin}\left(\frac{-a_x}{g}\right) \quad (1)$$

In (2), a_y, is the acceleration in the acceleration in the Y axis, a_z is the acceleration in the Z axis, and atan2 is a trigonometrical function. This equation gives us R, the roll angle.

$$R = \text{atan2}\left(\frac{a_y}{a_z}\right) \quad (2)$$

In (3), m_y and m_x are the sensor's magnetometer readings in the Y and X axis, and atan2 is a trigonometrical equation. This formula gives us Y, the Yaw orientation angle.

$$Y = \text{atan2}\left(\frac{m_y}{m_x}\right) \quad (3)$$

After using the (1),(2) and (3), it's used a Kalman filter to do a sensor fusion using the already computed Euler angles and the 3 axis gyroscope data to smooth the angles data [16-17].

B. Walker Elevation

To get the patient's walker elevation during a session, it is acquired first the sensor's distance to the floor with the walker on the ground, and then, every time a reading is done, it subtracts that distance to get the elevation.

C. Force Calculus

To get the force done by the patient in each walker's feet, a calibration session was made with a weighing-machine to correlate the 4 sensor's voltage output to the Kilograms force performed and get the (4), (5), (6) and (7) characteristic equations for the walker feet.

$$F_{FLF} = 15{,}359 * x - 4{,}5061 \quad (4)$$

$$F_{FRF} = 15{,}411 * x - 5{,}4966 \quad (5)$$

$$F_{BRF} = 14{,}428 * x - 0{,}2549 \quad (6)$$

$$F_{BLF} = 15{,}374 * x - 2{,}0648 \quad (7)$$

In these force equations (4-7), the x stands for read voltage, F_{FLF} stands for force applied in the front left foot, F_{FRF} stands for force applied in the front right foot, the F_{BRF} stands for the force applied in the back right foot, and the F_{BLF} stands for the force applied to the back left foot. Finally, the output of each this equations it's in Kg.

D. Center of pressure calculus

To acquire the center of the forces in each walker foot, it were used (8) and (9), which correlates the force applied to each walker's foot with his x and y coordinate, relative to the walker in a division between sums to give the centroid of the exerted forces (Fig. 7).

$$Pcx = \frac{\sum_{k=1}^{k=4} F_k * X_k}{\sum_{K=1}^{K=4} F_k} \quad (8)$$

$$Pcy = \frac{\sum_{k=1}^{k=4} F_k * Y_k}{\sum_{K=1}^{K=4} F_k} \quad (9)$$

Furthermore, F_k stands for the force applied to each foot, and the Y_k and X_k stands for each foot Y and X coordinate position.

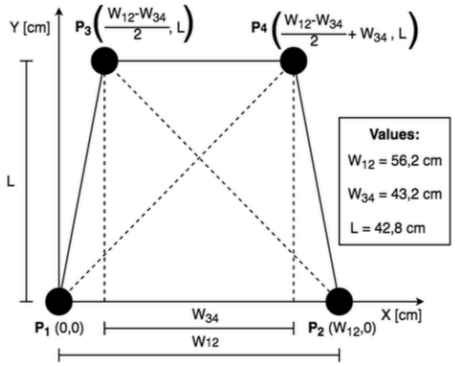


Fig. 7. Walker's diagram with the feet coordinates.

IV. Results and Discussion

Once the current paper presents a physical rehabilitation system proposal, with the focus being the experimental data validation and consistency regardless of repetition, the experiments were performed by a small set of healthy volunteers. That volunteer set was composed by 5 healthy males, with ages between 24-30 years, weights between 65-80 kg and heights between 1.70-1.80 m.

The sessions consisted in doing a ten consecutive steps walk using the smart walker. In order to test the system, 3 of the volunteers were asked to simulate to have lateral side body impairments. The other 2 were asked to perform the session without pretending to have any disabilities.

Those impairments would theoretically affect the elevation of the walker by the volunteer, elevating one side more than the other, and his center pressure, causing a deviation to the affected side.

Fig. 8 compares the elevation of the walker when the volunteer has some kind of right body side impairment (top) to the walker's elevation when the volunteer does not have any disability (bottom). Additionally, this figure shows that the elevation done by the volunteer walking normally has little variation between the right and left elevation of the walker, and that in opposition, the volunteer with the lateral impairment performed a higher elevation in the opposite side of his

impairment (lifting in average 6 more centimeters than the other side), which can be related to lack of strength.

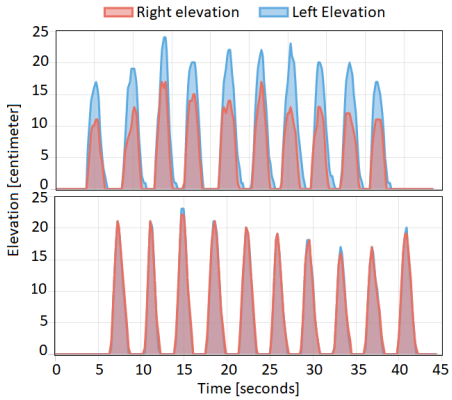


Fig. 8. Abnormal left side walker elevation (top); normal walker elevation (bottom).

Fig 9, 10 and 11, shows each one 4 types of center pressure patterns, acquired during the sessions. In the plots, the blue curve represents the top view of the walker, and the red points, the instant center pressure of the applied forces.

Fig 9 presents 4 types of left side deviated center pressure, performed by a volunteer simulating a left side impairment. As can be seen in this figure, the impairment leads to a higher pressure on the left side of the walker, resulting in a left side center pressure deviation. Additionally, this figure shows two plots were the impairment seems to be more severe (bottom-left and top-right plots), having the center pressure points more concentrated in the front left leg of the walker, and another 2 were the volunteer's physical condition is better (bottom right and top left plots), once the center pressure points (COP) are more disperse.

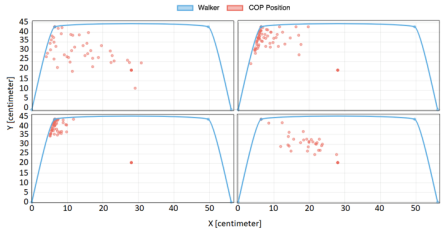


Fig. 9. Simulation of the pressure center deviated to the left.

Fig 10, shows 4 cases of anomalous gait (right side deviated center pressure) performed by a volunteer simulating a right side impairment. This figure verifies that having an impairment on the right side of the body leads to a higher pressure on the affected side, and consequently to a center pressure points deviated to the right side of the walker. Additionally the figure presents a more severe impairment case (top-left plot), once it concentrates the center pressure point on the walker structure, and 3 cases where the impairment condition is better, having the center pressure points more disperse and deviated from the walker structure.

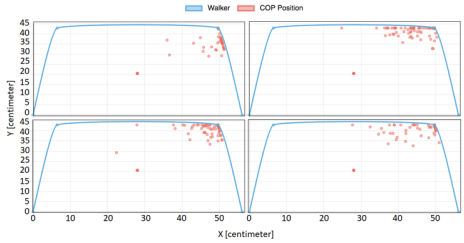


Fig. 10. Simulation of the pressure center deviated to the right.

Fig 11 presents 4 cases of normal center pressure made by 2 volunteers. The 4 cases show center orientated disperse set of center pressure points. Additionally, the figure shows that although the center pressure points are disperse, they are deviated from the sides of the walker structure.

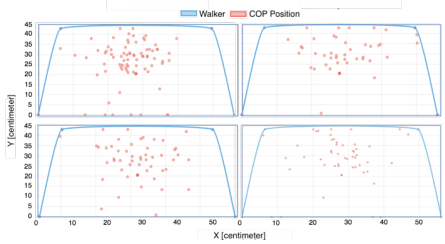


Fig. 11. Simulation of the normal pressure center.

V. Conclusions and Future Work

An elderly physical rehabilitation IoT solution to improve rehabilitation conditions, to provide more precise diagnostics, and consequently lower the recovery time was presented in the paper. The research and development of these type of physiotherapy systems, are important since the medicine enhancements are leading us towards an aging population, with less mobility, less independence and with increased healthcare needs.

System's such as the presented in the current paper, that measures, shows in real time the acquired data, and that stores patient's center pressure data, bilateral walker elevation, the orientation angles and the number of steps taken during a session in a remote server are important, as the results show, to archive better patient's performance analysis, and consequently

reach better mobility range, improve patient's strength and prevent patient's falls.

The results obtained in the performed experiments showed the real potential of the developed system to provide better quality of life, range mobility improvements and fall prevention reduction, that can lead to new impairments, to the users of walkers.

In the future the goal will be to use IoT big data analytics applied for physical rehabilitation, to develop rehabilitation and patient evolution prediction models based on the acquired data, during the sessions, as well as to improve the system's autonomy, and to adapt the described system to other physiotherapy tools.

VI. Acknowledgments

This research is supported by Instituto de Telecomunicações (IT-IUL) at ISCTE-IUL, Lisbon, Portugal and Fundação para a Ciência e Tecnologia (FCT) project PTDC/DTP-DES/6776/2014.

References

[1] World Health Organization (2011). World Report on Disability. [online] Malta, pp.22-40. Available at: http://www.who.int/disabilities/world_report/2011/report.pdf?ua=1 [Accessed 7 Feb. 2018].

[2] World Health Organization, "Rehabilitation 2030: A Call for Action", 2017.

[3] United Nations, Department of Economic and Social Affairs, Population Division (2015). World Population Ageing 2015 (ST/ESA/SER.A/390). Available at: http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2015_Report.pdf [Accessed 3 Feb. 2018].

[4] M. Manty, A. Heinonen, A. Viljanen, S. Pajala, M. Koskenvuo, J. Kaprio and T. Rantanen, "Outdoor and indoor falls as predictors of mobility limitation in older women", Age and Ageing, vol. 38, no. 6, pp. 757-761, 2009.

[5] World Health Organization, "WHO Global Report on Falls Prevention in Older Age", 2007.

[6] N. Peel, "Epidemiology of Falls in Older Age", *Canadian Journal on Aging / La Revenue canadienne du vieillissement*, vol. 30, no. 1, pp. 7–19, 2011.

[7] Y. Jiang, Y. Qin, I. Kim and Y. Wang, "Towards an IoT-based upper limb rehabilitation assessment system", *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seogwipo, 2017, pp. 2414-2417.

[8] G. E. De Luca, E. A. Carnuccio, G. G. Garcia and S. Barillaro, "IoT fall detection system for the elderly using Intel Galileo development boards generation I", *2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, Buenos Aires, 2016, pp. 1-6.

[9] O. Postolache, J. M. D. Pereira, V. Viegas and P. S. Girão, "Gait rehabilitation assessment based on microwave Doppler radars embedded in walkers", *2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings*, Turin, 2015, pp. 208-213.

[10] O. Postolache *et al*., "Smart walker solutions for physical rehabilitation" in *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 5, pp. 21-30, October 2015.

[11] L. Vogt, K. Lucki, M. Bach, and W. Banzer, "Rollator use and functional outcome of geriatric rehabilitation", J. Rehabilitation Res. Dev., vol. 47, no. 2, pp. 151-156, 2010.

[12] O. Postolache, "Physical rehabilitation assessment based on smart training equipment and mobile APPs", *2015 E-Health and Bioengineering Conference (EHB)*, Iasi, 2015, pp. 1-6.

[13] R. Gerena *et al*., "Wireless instrumented walker for remote rehabilitation monitoring", *2015 IEEE Virtual Conference on Applications of Commercial Sensors (VCACS)*, Raleigh, NC, 2015, pp. 1-7.

[14] T.Jesus and I.Silva, "Toward an evidence-based patient-provider communication in rehabilitation: linking communication elements to better rehabilitation outcomes", Clinical Rehabilitation, vol. 30, no. 4, pp. 315-328, 2015.

[15] "IDC: Smartphone OS Market Share", IDC: The premier global market intelligence company, 2018. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os. [Accessed: 09-Apr-2018].

[16] "How a Kalman filter works, in pictures | Bzarg", *Bzarg.com,* 2015. [Online]. Available: http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/. [Accessed: 17-Mar-2018].

[17] "Kalman filter vs. complementary filter", Robottini, 2011. [Online]. Available: http://robottini.altervista.org/kalman-filter-vs-complementary-filter. [Accessed: 17-Mar-2018].

2º artigo: Physical Rehabilitation based on Smart Walker

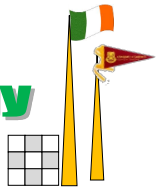
Este artigo foi aceite e será apresentado na 12ª edição do *ICST* (*International Conference on Sensing Technology*), uma conferência organizada pela *University of Limerick* em cooperação com os *chapters* do Reino Unido e da Irlanda do *IEEE Instrumentation & Measurement Society*, nos dias 3 a 6 de Dezembro de 2018, na *University of Limerick*, na cidade de Limerick, na Irlanda.

ICST'18

12th International Conference on

Sensing Technology

Dec. 3rd to 6th, 2018. Limerick, Ireland.



International Advisory Committee
I. Collings, MQ, Aus
S. Ueno, KU, Japan
E. Petriu, UO, Canada
K.T.V Grattan, CUL, UK
D.P. Tsai, AS, Taiwan
P. Sallis, AUT, NZ

General Chairs
T Newe, UL, Ireland
S. Mukhopadhyay, MQ, Aus

Technical Program Chairs
W Lyons, DKIT, Ireland
O. Postolache, ETA, Portugal
S. Kharkivskiy, WSU, Aus
A. Sanagavarapu, UTS, Aus
K.P Jayasundera, MU, NZ

Regional Programme Chairs
America: G. Chattopadhyay, JPL, USA
Eupore: I. Matias, PUN, Spain
Middle-East: C. Gooneratne, KAUST
Asia: R. Huang, NCKU, Taiwan
A. Mason, LJMU, UK
D. Preethichandra, CQU, Australia

Publicity Chair
K. Kuo, IIIR, Taiwan

International Programme Committee
E. Sazonov, UA, USA
Rajan Shankaran, MQ, Aus
I. Ihara, NUT, Japan
M. Yuce, MU, Australia
L.Tang, MU, NZ
S. Bhadra, McGill Uni, Canada
I. Woodhead, LT, NZ
Y.C.Liang, NUS, Singapore
W.Y. Chung, PKNU, Korea
J. Kosel, KAUST, SA
H. Ewald, Rostock Uni., Germany
V. Sberveglieri, CNR-IBBR, Italy
G.Liu, MQ, Aus
M.A. Yunus, UTM, Malaysia
R. Dykstra, VUW, NZ
N.K.Chen, LU, China
B. George, IITM, India
S. Yamada, KU, Japan
Y.H. Chung, PKNU, Korea
E. Lewis, Uni. Of Limerick, Ireland
C. Alippi, PM, Italy
M. Haji-Sheikh, UNI, USA
T. Okayasu, KU, Japan
C.H.Chuang, STU, Taiwan
K. Rajanna, IISc, India
H. Leung, UC, Canada
T.Newe, UL, Ireland
A. Flammini, UniBs, Italy
M. Nadi, U de Lorraine, France
R. Yan, SU, China
T. Bosch, TU, France
K. Tashiro, SU, Japan
V.J. Kumar, IITM, India
S. Ikezawa, WU, Japan
H. Ewald, U Rostock, Germany
T. Sun, CU London, UK
J.A. Jiang, NTU, Taiwan
A. Fuchs, GUT, Austria
T. Azuma, UU, Japan
I. Platt, LT, NZ
J. Zou, CHHK, HK
M. Soleimani, UB, UK
P. Arpaia, US, Italy
J.E.Y. Lee, CUHK, HK
C.S. Kim, MUST, USA
L. Rufer, CNRS, France
K. Shimanoe, KU, Japan
H. Wakiwaka, Shinshu Uni., Japan
Anoop C.S. IITKGP, India
And many others

Call for Papers

The University of Limerick, Ireland is pleased to announce that the 12th International Conference on Sensing Technology (ICST 2018) will be held on December 3rd to 6th, 2018 in Limerick, Ireland. ICST 2018 is intended to provide a common forum for researchers, scientists, engineers and practitioners throughout the world to present their latest research findings, ideas, developments and applications in the area of sensing technology. ICST 2018 will include keynote addresses by eminent scientists, Student Workshops as well as special, regular and poster sessions. All papers will be **peer** reviewed on the basis of a full length manuscript and acceptance will be based on quality, originality and relevance. Accepted papers will be published in the conference proceedings with an IEEE catalog number and ISBN number. **The proceedings will be submitted for publication in IEEE Xplore and suitable Extended papers will be invited for a special issue in Sensors (MDPI-Open Access).**

Topics will include, but are not limited to, the following:

1. Vision Sensing
2. Sensors Signal Processing
3. Sensors and Actuators
4. Sensors Phenomena and Modelling
5. Sensors Characterization
6. Smart Sensors and Sensor Fusion
7. Electromagnetic Sensors
8. Chemical and Gas Sensors
9. Physical Sensors
10. Electronic Nose Technology
11. Biological Sensors
12. Electro-optic Sensors and Systems
13. Mechanical sensors (inertial, pressure, and tactile)
14. Nano Sensors
15. Acoustic, Noise and Vibration Sensors
16. Wireless Sensors and WSN
17. Body Area Network
18. Internet of Things (IoT)
19. Security and Reliability of WSN
20. Optical Sensors (radiation sensors, optoelectronic/photonic sensors, and fibres)
21. Lab-on chip
22. Sensor Arrays
23. Intelligent sensing
24. Telemetering
25. Online monitoring
26. Applications of Sensors (automotive, medical, environmental monitoring, earthquake life detection, high speed impact, consumer, alarm and security, military, nautical/marine, aeronautical and space sensor systems, robotics, automation and manufacturing)
27. Solid State Sensors
28. Sensors for high energy physics applications
29. Particle accelerators and detectors
30. Internet-based and other Remote Data Acquisition
31. Education using sensors

Paper Submission

Authors are invited to submit the full manuscript (4 to 6 pages including references) of their technical paper, for oral or poster presentation. Papers in pdf format may be uploaded via the web using EDAS paper management system at **www.edas.info.**
The MS Word and pdf template will be available on the conference website. Final manuscripts are limited to **six** (6) A4 size pages.
In submitting a paper, the author(s) agree that, upon acceptance, they will prepare the final manuscript in time for inclusion in the published proceedings and will present the paper at the conference.

NOTE: The final manuscript will not be published without advance registration.

Special Sessions

ICST 2018 solicits special session proposals. The special sessions are intended to stimulate in-depth discussions in special areas relevant to the conference theme. The session organizers will coordinate the review process for their session papers. The conference proceedings will include all papers from the special sessions. **Authors contributing to special sessions are required to register for the conference.** Please contact the general chair if you would like to organize a special session.

For further details, please contact: **Thomas Newe or Subhas Mukhopadhyay at ICST2018Chair@gmail.com**

Manuscript Submission	~~30th June 2018~~
	Extended to 21st July 2018
Acceptance Notification	30th August 2018
Camera Ready Submission	30th September 2018
Advance Registration	30th September 2018

Web site: http://www.ece.ul.ie/ICST2018

Hosted by: University of Limerick, Limerick, Ireland

Financial sponsors: University of Limerick; Fáilte Ireland National Tourism Development Authority; Meet in Ireland

Technical sponsors: IEEE; IEEE Instrumentation & Measurement Society; sfi Science Foundation Ireland; sensors an Open Access Journal by MDPI

O e-mail de aceitação do artigo por parte da organização da conferência é apresentado abaixo:

[ICST2018] Your paper #1570485303 ('Physical Rehabilitation based on Smart Walker')

EM

EDAS Conference Manager <help@edas-help.com> em nome de icst2018Chair@gmail.com

qui 30-08, 17:06

Miguel Nave; Yongsheng Yang <yangys_smu@126.com>; Octavian Adrian Postolache <opostolache@lx.it.pt>; icst2018Chair@gmail.com

Responder a todos

Dear Mr. Carlos Nave:

Congratulations - your paper #1570485303 ('Physical Rehabilitation based on Smart Walker') for ICST2018 has been accepted.

Your paper has been accepted in the following category: To be determined Later.

Physical Rehabilitation based on Smart Walker

Carlos Nave
ISCTE-IUL, Lisbon
Portugal,
cmara1@iscte-iul.pt

Yongsheng Yang
Shanghai Maritime University
Shanghai 201306, China,
yangys_smu@126.com

Octavian Postolache
Instituto de Telecomunicações, ISCTE-IUL, Lisbon, Portugal
opostolache@lx.it.pt

***Abstract*—This paper describes a smart walker based on Arduino Mega computation platform and multiple sensing channels to materialize a Physical Rehabilitation Internet of Things System. The system extracts metrics such as orientation, the number of steps performed, bilateral walker elevation and the patients balance using Inertial Measurement Units, ultrasound and Load Cell sensors. Such metrics are stored in a remote server and available through a system's web and mobile app.**

Keywords—Physical Rehabilitation; IoT; Sensors; Smart Walkers; Arduino.

I. Introduction

Year after year society, through medicine, manages to overcome health challenges and increase living conditions, making life expectancy rise considerably. This growth is contributing to an aging society. In fact, it's expected that by 2050 the number of people over 60 years of age will double the current values [1].

As times go by, the human being loses physical capabilities such as muscle strength such as muscle strength, elasticity or balance, which can lead to mobility and independence loss, accidents and falls. In fact, 20% to 40% of individuals over 65 fall every year, and these can lead to progressive reduction of mobility and the appearance of new walking complaints [2],[3].

The big growth of elderly population, in the medium-term, presents by itself a new challenge for healthcare system in general, since will be necessary to assure a good quality of services for lower prices that implies to promote new solutions in different healthcare services including smart physiotherapy.

The fast development of sensors and networking technologies and the development in the Internet Of Things field (IoT), have promoted the development of new healthcare solutions that plenary answer to different society challenges. These healthcare solutions enable interoperability and data integration between medical devices, mapping session's history, measurement error reduction, real-time monitoring and adaptive treatment to patient's needs.

The real-time monitoring feature can be used to evaluate the treatment progress, giving feedback to both patient and physiotherapist. Thus the communication motivation between patient and physiotherapist improves, enabling the patient to archive better and faster results, reducing time and treatment cost [4],[5].

In order to proceed to the system implementation, it was used an Arduino Mega, to collect the different types of signals (Analog and Digital) were developed scripts in C to acquire the sensors data so as PHP scripts to store the data on the remote server database. It was also developed a webapp and an Android App to display the session's history.

II. Related Work

There has been research and system's development in the physical rehabilitation sector to provide the professionals better diagnosis and to obtain better results. In that sense, there's a paper that describes a live progress evaluation IoT system using a piezoresistive matrix, a microcontroller and Wi-Fi connection [6]. In another paper a plantar pressure distribution IoT monitoring system using FSR sensors is described [7]. There's even a paper that presents a upper limb physical rehabilitation IoT system using EMG, motion and temperature sensors combined with machine learning to evaluate and classify stroke survivors physical condition [8].

Walkers and crutches are vastly used and accepted during rehabilitation sessions, giving it's lower risk of fall. In fact, there are already papers that describe the development of *IoT* Systems using these aids as a network node.

There are papers that describe the use of force sensors, IMU and microwave Doppler sensors mounted on walkers and crutches, to acquire metrics like gait patterns, orientation and force done by the patient in the walker's or crutches feet and palm rest, that are stored in a server and that can be accessed using the system's mobile App [9],[10].

In another paper, it's described the development of a low-cost real-time patient's gait evaluation system that uses ADS (Active Depth Sensor) implemented in walkers. The system has a feet tracking algorithm to determine walking patterns, and the distance between the patients feet and the walker, in order to determine a possible patient "drag" [11].

III. System Description

The system is mainly expressed by a smart walker physical rehabilitation system, that includes an Arduino Mega embedded computation platform, that calculate metrics based on the acquired signals from sensors channels, and sends the processed information to a server using a Wi-Fi module. Additionally a Bluetooth communication is used to deliver the data to a mobile device as it seen in Fig. 1.

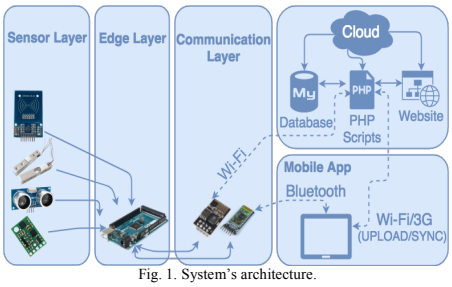


Fig. 1. System's architecture.

The developed system is characterized by hardware component (smart walker) and software (Server, node and mobile software) part described in the A and B subsections.

A. Hardware

The system's physical part, the smart walker, is powered by an Atmel Atmega 2560 microcontroller, the edge layer of the system (Fig. 1), and has sensors, a signal conditioning circuit and two communication modules attached to him. The present sensors are a RFID reader (1), an IMU sensor (2), four load cell sensors (3), two ultrasound sensors (4) , and the communication modules are a (5) Wi-Fi and (6) Bluetooth modules as can be seen in Fig. 2.

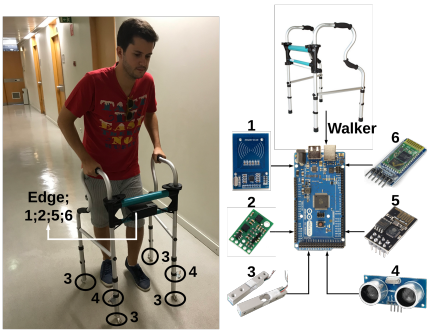


Fig. 2. Smart Walker Prototype; Smart Walker node diagram (right) containing (1) RFID reader, (2) IMU Sensor, (3) load cells, (4) ultrasound sensors, (5) Wi-Fi and (6)Bluetooth modules.

• Sensors

1) RFID Reader

This 13.56 MHz RFID sensor is able to read passive RFID tags/cards, activating his passive circuit when they are 50mm apart from each other. This sensor is used as a patient identifier, at the start of a session.

2) IMU

This 9DoF IMU, has an accelerometer, a gyroscope and a magnetometer, with 3 axis each one, and it's widely used as a motion detector in GPS navigation, VR and robotics. It's used the I2C protocol, and it's acquired the patient's orientation with the Euler angles using Kalman Filters.

3) Micro Load Cell

This sensor measures the force in one direction, and has a [0;500] N sensibility interval. Given it's too week output it's used a signal conditioning circuit that amplifies the signal to a [0;4]V interval. The walker has 4 of these to measure the force in each walker feet and the patient's gait. There were performed calibrations in each sensor to get each characteristic curve.

4) Ultrasound sensor

This 5V sensor measures the distance between itself and an obstacle, in a range between [2-400]cm, using the time elapsed between the emission and reception of an ultrasound. There are 2 of these sensors, and they are used to measure the elevation of the walker from the ground and the number of steps taken.

• Communication Modules

5) WI-FI Module

This 3.3V module has a range of 90 meters, and connects the microcontroller to the internet through AT commands. It was used to store the metrics accessing PHP scripts stored in the remote server.

6) Bluetooth Module

This 10 meters range module, enables the wireless communication between electronics through Serial.
It's used to send the acquired metrics, during the session, to a mobile device with the system's App, to display in real-time the live data.

B. Software

The system's software is divided in three parts. The edge, the cloud and the app and webapp.

1) Edge Software

The edge software is the one that runs on the Atmel Atmega2560 microcontroller. This software is written in C language and is responsible for acquire the sensors raw data, to get the desired metrics with that data, and to send them to the system's cloud. The sending process can be done through the mobile app or directly depending on the type of session (Fig. 1). In the first case, the session type is "real time session", and the data goes from the edge layer to the mobile app, through the Bluetooth communication module, showing real time data at the same time that sends it to the cloud using the already described PHP scripts. In the second case, the data goes directly from the smart walker's Wi-Fi module to the cloud using PHP scripts.

2) Cloud Software

The cloud software is divided in 2 sections. The system's database, the system's webapp and the needed PHP script files. The system's database is a 9 tables MySQL database that stores the user's personal data (Admin, Physiotherapist and Patient tables), the patient's sessions data (Session, Orientation, Elevation and Pressure tables) and the patient's workouts data (Workout and Exercise tables), as can be seen in Fig. 3.

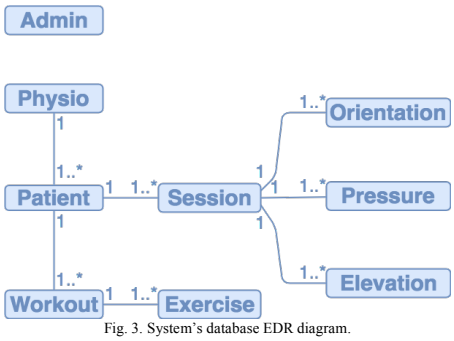


Fig. 3. System's database EDR diagram.

The PHP script files are used so the Edge level, the Mobile app and the to get the required data to populate the webapp and the mobile app, or to insert the acquired data by the edge layer of the system.

3) User interfaces

The registered users (administrators, physiotherapists or patients) have a webapp and an Android mobile application so they can see all the system's data, as can be seen in Fig. 4.

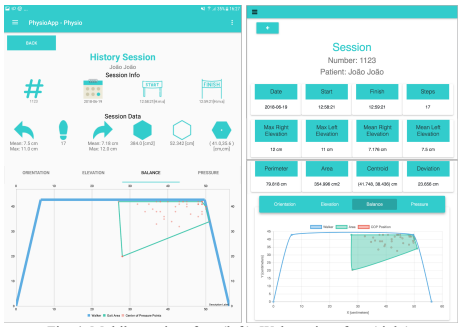


Fig. 4. Mobile app interface (left); Webapp interface (right).

The webapp is a responsive one, which means that is adaptable to every screens sizes and resolutions, and it was developed for its cross platform ability.

On the other hand the the mobile app was developed for Android devices since it's been the users preference and registered a 85% smartphone market share in 2017 [12].

These two interfaces use a set of PHP script files stored in the cloud server so they can interact with the system's database and receive or insert data from/into the system. To access the PHP files the interfaces use HTTP POST/GET requests.

Depending on the user type, different actions can be performed, as showed in Fig. 5.

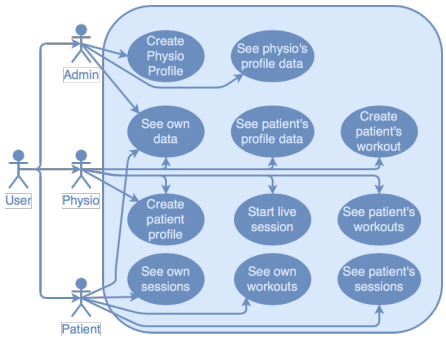


Fig. 5. System's user actions.

IV. Sensing Algorithms

This section presents the calculus done in the network node to acquire the desired metrics so they can be stored in the cloud server afterwards.

A. Orientation angles calculus

To obtain the Euler angles (roll, pitch, and yaw), were used (1), (2) and (3) equations. These equations relate the accelerometer, magnetometer and gyroscope data acquired from the IMU sensor, to obtain the desired angles.

In (1), P is the pitch angle, a_x stands for the accelerometer value in the X axis, g stands for the gravitational constant, and asin is a trigonometrical function.

$$P = \text{asin}\left(\frac{-a_x}{g}\right) \quad (1)$$

In (2), R is the roll angle, a_y, stands for the accelerometer value in the Y axis, g is the gravitational constant, and atan2 represents arctan, a trigonometrical function.

$$R = \text{atan2}\left(\frac{a_y}{a_z}\right) \quad (2)$$

In (3), Y is the yaw angle, m_x and m_y stand for the X and Y axis components of the magnetometer obtained value, and atan2 represents a trigonometrical function.

$$Y = \text{atan2}\left(\frac{m_y}{m_x}\right) \quad (3)$$

To get more stable output angles values it's used a Kalman filter. This filter uses the gyroscope readings and the time elapsed between readings to soften the angles [13].

B. Walker elevation and steps calculus

To obtain the walker elevation during a session, first it's used (4) to calculate the sensor's distance to the ground when the walker is standing. In this equation *d* represents the acquired distance in cm, *t* is the time elapsed between the transmission and reception of the ultrasonic wave, transmitted by the sensor, in seconds, and v_s the speed of sound in cm/s.

$$d = \left(\frac{t * v_s}{2}\right) \quad (4)$$

After that, it's calculated the elevation of the walker by subtracting the already calculated offset to (4).

The steps count is obtained by incrementing one unit every time the walker elevation goes from zero to one centimeter.

C. Walker feet force calculus

To calculate the forces applied to the feet of the walker, it was done a calibration session with a weighing-machine to associate the sensors output voltage to the force applied in N. That calibration, gave the (5), (6), (7) and (8) characteristic equations were obtained.

$$F_{FLF} = 153{,}59 * x - 45{,}061 \quad (5)$$

$$F_{FRF} = 156{,}9 * x - 57{,}786 \quad (6)$$

$$F_{BRF} = 153{,}74 * x - 20{,}648 \quad (7)$$

$$F_{BLF} = 144{,}28 * x - 2{,}5486 \quad (8)$$

In (5),(6),(7) and (8), x stands for the sensor's output voltage, F_{FLF}, F_{FRF}, F_{BRF} and F_{BLF} stands for the force applied to the front left foot, to the front right foot, to the back right foot and to the back left foot, respectively. These equations receive each sensor output value in volts (x), and give each sensor's applied force in N.

D. User's center pressure calculus

The walker's center pressure is given by (9) and (10) equations. These equations relate the force applied to each walker foot, in N, to their x and y position coordinates, in cm, relative to the walker in a division between sums, as can be seen in Fig 7.

$$Pcx = \frac{\sum_{k=1}^{k=4} F_k * X_k}{\sum_{K=1}^{K=4} F_k} \quad (9)$$

$$Pcy = \frac{\sum_{k=1}^{k=4} F_k * Y_k}{\sum_{K=1}^{K=4} F_k} \quad (10)$$

In (8) and (9) F_k represents the force applied to each walker foot, Y_k and X_k represents the X and Y coordinates of the walker feet, and *Pcx* and *Pcy* stands for the X and Y coordinates of the center pressure centroid.

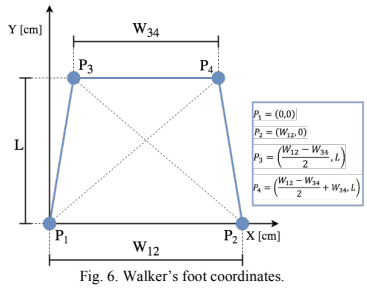


Fig. 6. Walker's foot coordinates.

V. Results and Discussion

An experiment session was created and performed by 3 healthy volunteers. Two of the volunteers had the objective of emulate an impairment on one side of their bodies (one on the left side and the other one on the right side). The other volunteer performed the exercises normally, without having to emulate to have an impairment.

Theoretically, a person with side body impairment could present a higher elevation of the walker opposite side of his impairment, due to lack of strength, as well as a deviated body center pressure to his impairment side.

The session consisted In having two exercises done. The first one was a 5 meters straight line walk, and the second one a 5 meters zigzag walk.

Regarding the volunteers center of pressure, Fig. 7 and Fig. 8 shows the volunteers performance in the first exercise, and second exercise, respectively.

In Fig. 7 the volunteers with an side body impairment had a center pressure deviated to their impairment side, and that the other volunteer had a center pressure centered due to lack of injuries.

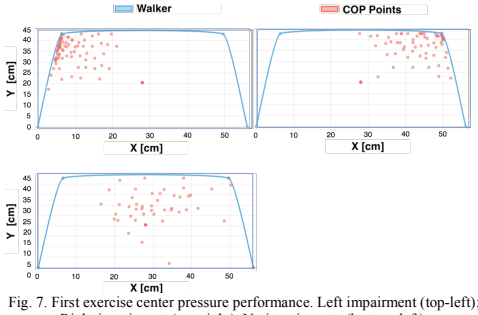


Fig. 7. First exercise center pressure performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

In Fig. 8, although the performance pattern seems to be the same, it can be seen that the center of pressure points are more concentrated in the exercise execution of the three volunteers. That shows that the injured volunteers had a worse and aggravated performance doing the second exercise, that is more complex exercise, where they have to walk and rotate at the same time.

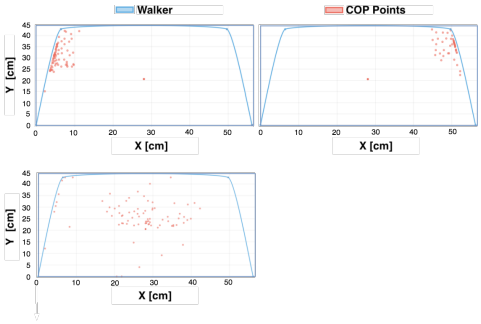


Fig. 8. Second exercise center pressure performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

The volunteers' walker elevation performance in the first and second session's exercises can be seen in Fig. 9 and Fig. 10, respectively.

In Fig. 9, its shown that the volunteers with injuries (left-side injurie in the top-left plot, right-side injurie in the top-left plot), had a higher elevation of the opposite side of their injurie. That could be associated with a lack of strength on the impairment body side, favoring the other side.

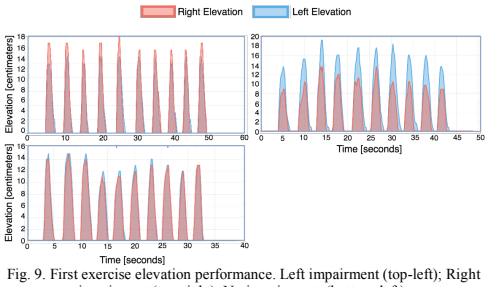


Fig. 9. First exercise elevation performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

In Fig. 10, it can be seen the same pattern, i.e., the higher walker elevation on the injurie opposite side, but it can also be noticed that the volunteers had a higher elevation side difference, possibly because the second exercise was harder due to torso rotations while walking, to perform a zigzag walk.

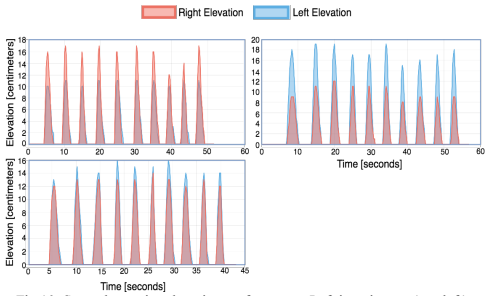


Fig.10. Second exercise elevation performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

Regarding the three volunteers orientation, Fig. 11 and Fig. 12 shows their performance in the first and second session's exercises, respectively.

In Fig. 11, it can be seen that the impaired volunteers (left-side injurie in the top-left plot, right-side injurie in the top-left plot) struggle to do the 5 meters walk keeping the direction straight. The injured volunteers oscillate between left and right directions, as can be seen in their yaw angle line plots. Regarding the volunteer with no impairments, it can only be seen some low oscillations, associated with normal movement.

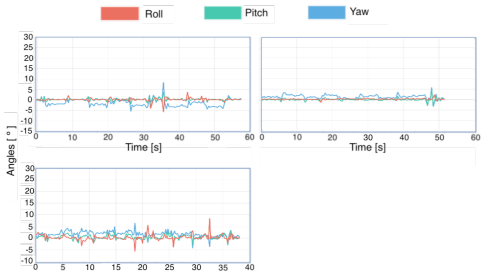


Fig. 11. First exercise orientation performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

Fig. 12 shows that the injured volunteers (left-side injurie in the top-left plot, right-side injurie in the top-left plot) had problems to rotate to the opposite side of their impairment in the second exercise. As can be seen, in the Yaw angle line dataset of those volunteers, they had a lower torso rotation to their impairment side, that can be associated with a unconscious movement to prevent a possible fall.

Furthermore, Fig. 12 also shows that the volunteer that doesn't emulate any impairment performs a similar rotation (yaw angle) in both sides.

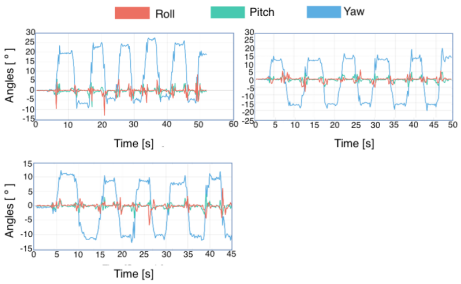


Fig. 12. Second exercise orientation performance. Left impairment (top-left); Right impairment (top-right); No impairments (bottom-left).

VI. Conclusions and Future Work

A low cost smart walker rehabilitation system is presented in the paper. The system acquires and stores important gait metrics, such as walker's elevation, walker's orientation angles, applied forces or center of pressure while a session is undergoing. The system also provides the comparison of the last 5 patient's sessions for evolution diagnosis purposes, and enables the physiotherapist to see the session's data in real time. The kind of session's data presented by the system is relevant and important since it can allow both physiotherapist and patient to have a better understanding of the patient's evolution stage so the treatment program can be more efficient, the recovery time can be shortened and the results can be better.

The obtained results from tests prove that the implemented prototype can be helpful to detect and later correct efficiently lack of arm's strength and lack of gait, as well as to determine upper body rotation range motion, so the patient can regain quality of life to do the day-to-day activities.

In the future the goal will be to use data mining tools to identify and classify gait patterns such as predominance of one side elevation, or balance deviation during a session or a longer treatment, as well as to use ECG sensors to measure the patient effort during a session.

Acknowledgments

This research is supported by Instituto de Telecomunicações (IT-IUL) at ISCTE-IUL, Lisbon, Portugal and Fundação para a Ciência e Tecnologia (FCT) project PTDC/DTP-DES/6776/2014 and by ISCTE/ IT-IUL – SMU Joint Laboratory.

References

[1] ONU, "World population, ageing," *Suggest. Cit. United Nations, Dep. Econ. Soc. Aff. Popul. Div. (2015). World Popul. Ageing*, vol. United Nat, no. (ST/ESA/SER.A/390, p. 164, 2015.

[2] M. Mänty *et al.*, "Outdoor and indoor falls as predictors of mobility limitation in older women," *Age Ageing*, vol. 38, no. 6, pp. 757–761, 2009.

[3] N. M. Peel, "Epidemiology of Falls in Older Age," *Can. J. Aging / La Rev. Can. du Vieil.*, vol. 30, no. 1, pp. 7–19, 2011.

[4] L. Vogt, K. Lucki, M. Bach, and W. Banzer, "Rollator use and functional outcome of geriatric rehabilitation," *J. Rehabil. Res. Dev.*, vol. 47, no. 2, p. 151, 2010.

[5] T. S. Jesus and I. L. Silva, "Toward an evidence-based patient-provider communication in rehabilitation: Linking communication elements to better rehabilitation outcomes.," *Clin. Rehabil.*, p. 0269215515585133-, 2015.

[6] M. Rossi, A. Rizzi, L. Lorenzelli and D. Brunelli, "Remote rehabilitation monitoring with an IoT-enabled embedded system for precise progress tracking," *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Monte Carlo, 2016, pp. 384-387.

[7] P. S. Malvade, A. K. Joshi and S. P. Madhe, "IoT based monitoring of foot pressure using FSR sensor," *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2017, pp. 0635-0639.

[8] Y. Jiang, Y. Qin, I. Kim and Y. Wang, "Towards an IoT-based upper limb rehabilitation assessment system," *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seogwipo, 2017, pp. 2414-2417.

[9] O. Postolache and P. S. Girão, "Physiotherapy smart connected devices for S-health," *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chalkidiki, 2016, pp. 1-6. doi: 10.1109/IISA.2016.7785336.

[10] O. Postolache *et al.*, "Smart walker solutions for physical rehabilitation," in *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 5, pp. 21-30, October 2015. doi: 10.1109/MIM.2015.7271223

[11] M. Martins, C. P. Santos, S. Page, L. Saint-Bauzel, V. Pasqui, and A. Meziere, "Real-Time gait assessment with an active depth sensor placed in a walker," *IEEE Int. Conf. Rehabil. Robot.*, vol. 2015–September, pp. 690–695, 2015.

[12] "IDC: Smartphone OS Market Share", IDC: The premier global market intelligence company, 2018. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os. [Accessed: 03-Feb-2018].

[13] "How a Kalman filter works, in pictures | Bzarg", *Bzarg.com*, 2015. [Online]. Available: http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/. [Accessed: 17-Mar-2018].

Apêndice B – Manual do utilizador

Instituto Universitário de Lisboa
Departamento de Ciências e Tecnologias da Informação

Manual do utilizador

IoPhyR – Internet of Physical Rehabilitation IoT System

Carlos Miguel Alpedrinha Ramos de Almeida Nave

Orientador(a):
Doutor Octavian Postolache, Professor Auxiliar,
ISCTE-IUL

Setembro, 2018

IoPhyR – Manual do utilizador

i

Índice

IoPhyR – Manual do utilizador

Índice de Figuras

Descrição do manual de utilizador

O presente manual de utilizador tem como objetivo expor o funcionamento das aplicações do sistema. No primeiro capítulo é abordada a aplicação embebida do sistema e como se processa a sua utilização. No segundo apresenta-se o funcionamento da aplicação móvel do sistema. Já no terceiro, identifica-se as funcionalidades e a correta utilização da aplicação *web*. Finalmente, no capítulo 4 expõe-se o manual de instalação das aplicações do sistema.

Capítulo 1 – Aplicação embebida

Neste capítulo apresenta-se a descrição e funcionamento da aplicação embebida desenvolvida para o protótipo de andarilho do sistema *IoPhyR*.

1.1. Descrição da aplicação

A aplicação embebida pertencente ao sistema *IoPhyR* foi desenvolvida para ser utilizada por pacientes com algum tipo de incapacidade motora, sob a supervisão dos seus fisioterapeutas. Este manual serve para dar a conhecer a aplicação e o seu correto funcionamento aos fisioterapeutas, de modo a que estes possam auxiliar os seus pacientes na sua utilização.

O protótipo desenvolvido, o *smart walker* ou andarilho, calcula métricas relativas à orientação e equilíbrio do paciente, assim como à elevação bilateral do andarilho, dos passos dados e das forças exercidas sobre os pés do mesmo. Em relação ao seu acesso, este está restringido a fisioterapeutas e pacientes registados no sistema, munidos de um cartão de paciente, e/ou um dispositivo móvel com a aplicação do sistema.

1.2. Funcionamento da aplicação

De modo a ser realizada uma correcta utilização do protótipo de andarilho desenvolvido, apresentam-se as seguintes acções que devem ser realizadas sequencialmente:

Ponto prévio: O andarilho necessita de uma conexão Wi-Fi onde se ligar para poder operar. A rede Wi-Fi deverá ter como nome “iPhone de Alberto” e password “tailorphy2018”.

Ligar o andarilho: Primeiramente deve ser ligado o andarilho. Para isso deve ser deslocado o interruptor presente no andarilho para a posição *ON* (ver Figura 1.1).

Figura 1.1 – Interruptor do andarilho.

Verificação do nível de energia: Em segundo lugar deve ser verificado o nível de energia do protótipo de andarilho. Para isso basta verificar se o *led* vermelho presente na unidade central do andarilho, colocado entre as barras frontais do andarilho se encontra ligado (ver Figura 1.2).

Figura 1.2 – *Led* indicador do nível de bateria do andarilho.

Em caso afirmativo poder-se-á passar para o próximo passo, a inicialização da sessão. Caso contrário, o andarilho terá ficado sem bateria, e terá de ser carregado na porta *mini USB* disponível na lateral esquerda da referida unidade central do andarilho, com um carregador *USB* de 5V e 2A (ver Figura 1.3).

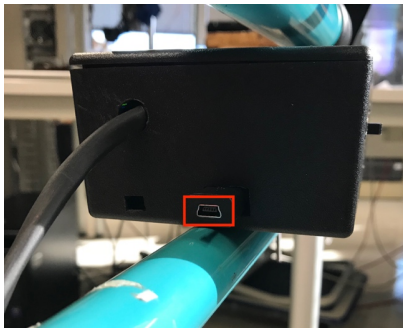


Figura 1.3 – Porta *mini USB* do andarilho.

Inicialização da sessão: Neste passo dá-se início à sessão de fisioterapia. Este pode ser feito das seguintes maneiras:

- **Usando o cartão do paciente:** Neste caso o paciente deve passar o seu cartão pelo leitor de cartões presente no protótipo. Este leitor está colocado na unidade central do andarilho, colocado do lado esquerdo e assinalado pelo termo "*Leitor*". O referido cartão deverá estar no máximo a 1 cm de distância do leitor, e deverá permanecer na zona de leitura no mínimo por 3 segundos (ver Figura 1.4).

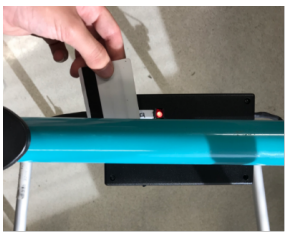


Figura 1.4 – Leitura de cartão do paciente.

Após o período de leitura, onde o cartão é verificado pelo sistema, uma de duas respostas será dada pelo mesmo utilizando os 2 *led*s existentes. No caso do cartão estar autorizado, ligar-se-à o *led* verde, e poder-se-á dar início a sessão, realizando os exercícios solicitados pelo fisioterapeuta (Figura 1.5). Por outro lado, caso o cartão não esteja no sistema, o *led* vermelho piscará 5 vezes, relatando que a sessão não poderá ser realizada (ver Figura 1.5).

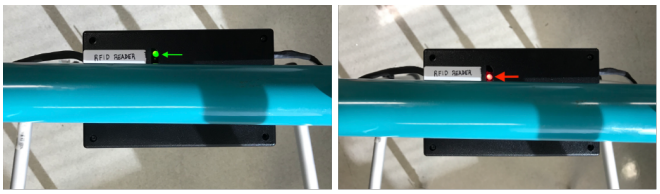


Figura 1.5 – Respostas dos *leds* à passagem do cartão do paciente.

- **Utilizando a aplicação móvel:** Para realizar a sessão de fisioterapia deste modo, o fisioterapeuta terá primeiramente de se fazer acompanhar de um dispositivo *Android* com a *PhysioApp*, e ter realizado o *login* bem-sucedido na aplicação. Depois terá de se dirigir à sua lista de pacientes, aceder ao perfil do paciente em causa e selecionar a opção de "*Real Time Session*". Lá poderá estabelecer uma comunicação *Bluetooth* com o protótipo de andarilho, clicando no botão "*START*" para visualizar as métricas recebidas em tempo real, e decidir quando terminar a sessão de fisioterapia. Deve ser confirmado ainda que o *led* verde presente na unidade central do andarilho se acendeu e o vermelho se apagou (ver Figura 1.6).

Nota: Esta comunicação é abordada mais detalhadamente na secção 2.1 do Capítulo 2 do presente manual.

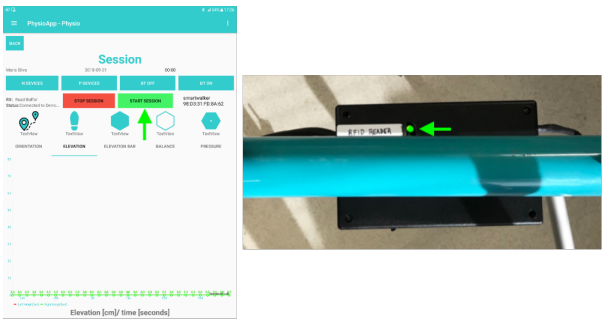


Figura 1.6 – Inicialização da sessão de fisioterapia na *PhysioApp*.

Término da sessão: Com este passo conclui-se a sessão de fisioterapia, estando o sistema disponível para começar uma nova sessão. No entanto e dependendo da forma como a sessão foi estabelecida esta terá de ser terminada de uma de duas maneiras:

- **Utilizando o cartão do paciente:** Caso tenha sido usado o cartão do paciente, o mesmo deverá ser usado para terminar a sessão. Sendo assim, este deverá ser passado novamente no leitor, previamente referido, a uma distância máxima de 2 cm, e por um período não inferior a 4 segundos. Deverá, no entanto, ser verificada a boa terminação da sessão verificando que o *led* vermelho está aceso, e o verde apagado (ver Figura 1.7).

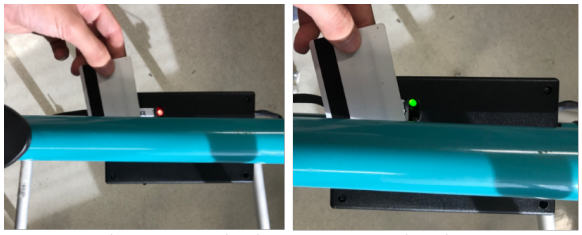


Figura 1.7 – Término da sessão com cartão do paciente.

- **Utilizando a aplicação móvel:** Caso tenha sido utilizada a *PhysioApp*, esta deve ser utilizada novamente para terminar a presente sessão. Nesse sentido, o fisioterapeuta premir o botão "*STOP*" presente na página "*Real Time Session*" aberta anteriormente para a realização da sessão de fisioterapia do paciente em causa. Esta acção realiza a comunicação com o protótipo de andarilho desenvolvido, terminando a sessão (ver Figura 1.8).

Nota: Esta comunicação é abordada mais detalhadamente na secção 2.1 do Capítulo 2 do presente manual.

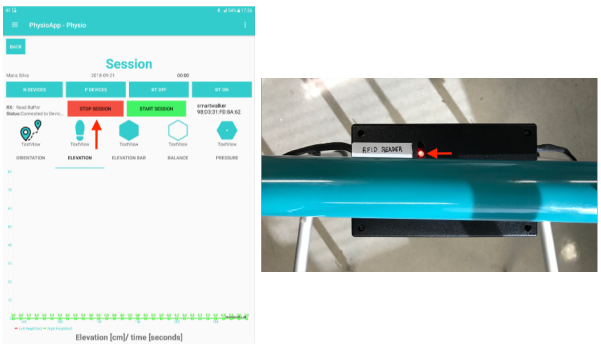


Figura 1.8 – Término da sessão de fisioterapia na *PhysioApp*.

Após a realização de cada sessão, os dados recolhidos são disponibilizados aos utilizadores registados no sistema, através da utilização das aplicações móvel e *web* do sistema.

IoPhyR – Manual do utilizador

6

Capítulo 2 – Aplicação *PhysioApp*

Neste capítulo apresenta-se a descrição e funcionamento da aplicação móvel do sistema *IoPhyR, a PhysioApp*.

2.1. Descrição da aplicação

A aplicação móvel pertencente ao sistema *IoPhyR*, a *PhysioApp*, foi desenvolvida exclusivamente para o sistema operativo *Android*. Sendo assim, poderá apenas ser instalada e utilizada por dispositivos que tenham esse sistema. No entanto, está disponível para os utilizadores com dispositivos com diferentes sistemas operativos, uma aplicação *web* que realiza funções semelhantes, a *PhysioWebapp*.

Nota: Para mais informações relativas à *PhysioWebapp* deve ser consultado o Capítulo 3.

Em relação aos requisitos técnicos do dispositivo onde será instalada a *PhysioApp*, este deve ter instalada pelo menos a versão do *Android Lollipop* (5.0), e dispor de acesso a *Internet* (*3G*, *LTE*, *4G*, *Wi-Fi*) e possuir *Bluetooth*.

Para utilizar a *PhysioApp*, o utilizador deverá ser um administrador, fisioterapeuta ou paciente e estar registado no sistema *IoPhyR*. Ao abrir a aplicação, é apresentada ao utilizador uma página de *Login* (ver Figura 2.1). Nessa página, deverá inserir as suas credenciais (*username* e *password*) nas *textfields* existentes para o efeito e clicar no botão de *Login*, logo abaixo.

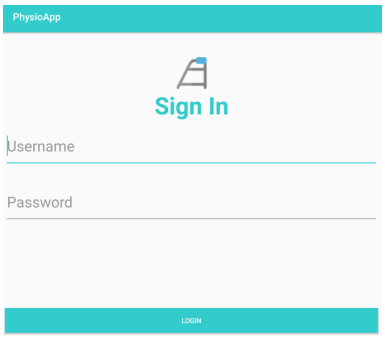


Figura 2.1 – Página de entrada da *PhysioApp*.

Caso o *login* seja bem-sucedido será apresentada uma mensagem de boas-vindas e o utilizador será deslocado para a sua página de perfil onde poderá realizar as acções que lhe são permitidas. Caso contrário será mostrada uma mensagem de erro com o texto "*Error*" (ver Figura 2.2).

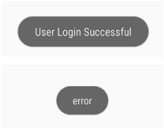


Figura 2.2 – Mensagens de entrada na *PhysioApp*.

Devido às particularidades de cada tipo de utilizador são apresentados 3 manuais, um para cada um deles.

2.2. Manual do administrador

Quando o administrador do sistema entra na *PhysioApp*, este é encaminhado para a sua página de perfil (ver Figura 2.3).

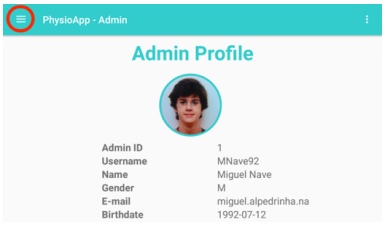


Figura 2.3 – Perfil do administrador na *PhysioApp*.

No seu perfil, o administrador pode visualizar as suas informações pessoais. como o seu *username*, nome, género, *e-mail* ou dia de aniversário. Clicando no botão marcado a vermelho na Figura 2.3, este acede ao menu lateral exibido na Figura 2.4.

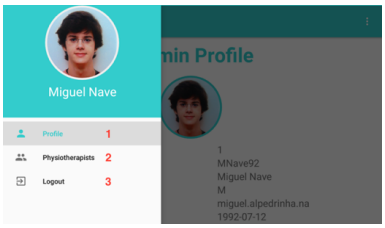


Figura 2.4 – Menu lateral do administrador na *PhysioApp*.

Neste o menu lateral, o administrador tem à sua disposição os seguintes 4 botões marcados com números de 1 a 4:

- **1** → Encaminha o administrador de novo para o seu perfil.
- **2** → Mostra a área dos fisioterapeutas.
- **3** → Realiza o logout do administrador na aplicação.

Caso o administrador decida clicar no botão número 2 do menu lateral mostrado na Figura 2.4, ser-lhe-á mostrada a página dos fisioterapeutas. Nesta página é possível ver a lista de fisioterapeutas existentes, sincronizá-la, ou pesquisar, criar ou selecionar um

fisioterapeuta (Ver Figura 2.5). Estas ações são descritas pormenorizadamente nas secções 2.2.1 a 2.2.4.

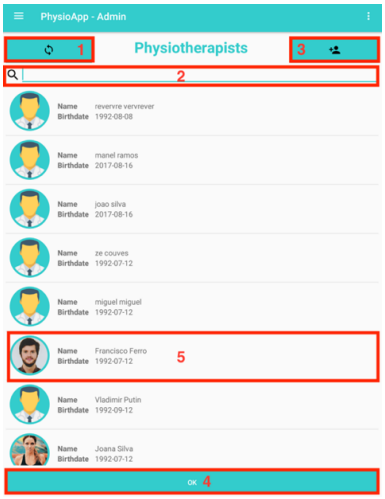


Figura 2.5 – Página dos fisioterapeutas na *PhysioApp*.

2.2.1. Sincronia de fisioterapeutas

Caso o administrador queira sincronizar a lista de fisioterapeutas, deverá clicar no botão assinalado com o número 1 na Figura 2.5. Se não existirem novos fisioterapeutas, será mostrada a mensagem apresentada no topo da Figura 2.6. De outra forma será mostrada a mensagem na base da mesma figura e os novos fisioterapeutas serão adicionados à lista existente.

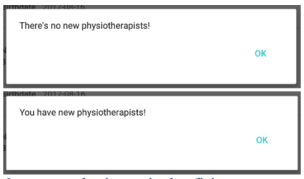


Figura 2.6 – Mensagens de sincronia dos fisioterapeutas na *PhysioApp*.

2.2.2. Pesquisar fisioterapeuta

O administrador tem a possibilidade de pesquisar um fisioterapeuta presente na lista. Para isso deve inserir no campo de texto marcado na Figura 2.5 com o número 2, alguma informação identificativa do fisioterapeuta em causa. Estes poderão ser encontrados pela sua data de nascimento, pelo seu nome, *username* ou *e-mail* (ver Figura 2.7).

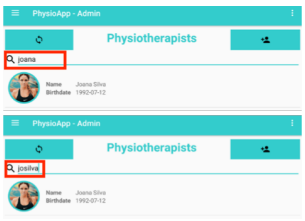


Figura 2.7 – Pesquisa por fisioterapeuta na *PhysioApp*.

2.2.3. Criar fisioterapeuta

Para criar um fisioterapeuta, o administrador deve clicar no botão marcado com o número 3 na Figura 2.5. Clicando nesse botão, é deslocado para a página de criação de fisioterapeuta, onde é apresentado um formulário que deverá preencher com as informações do fisioterapeuta (ver Figura 2.8). Isto é, a fotografia, o *username*, a *password*, o *e-mail*, o primeiro e último nome, o género e a data de nascimento do fisioterapeuta. Sendo que se pretender regressar à página dos fisioterapeutas deverá clicar no botão marcado com o número 1 na Figura 2.8.

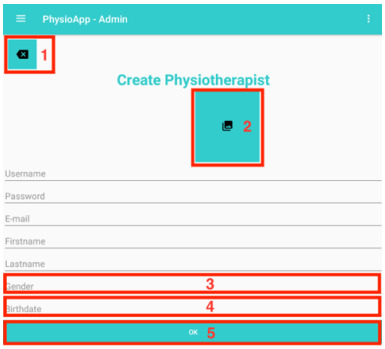


Figura 2.8 – Página de criação de fisioterapeuta na *PhysioApp*.

Ao clicar no botão número 2, é apresentado ao administrador o menu para adicionar uma fotografia mostrado na Figura 2.8. Nesse menu, o administrador pode tirar ou escolher na galeria do dispositivo uma fotografia para identificar o fisioterapeuta.

Figura 2.9 – Menu da fotografia do fisioterapeuta na *PhysioApp*.

Por outro lado, ao premir o botão número 3 da Figura 2.8, é mostrado ao administrador um menu de selecção do sexo do fisioterapeuta. Existem duas opções, *Male* ou *Female*,

podendo ser apenas escolhida uma das duas. Premindo uma das opções, o género do fisioterapeuta ficará identificado (ver Figura 2.10).

Figura 2.10 – Menu de selecção do género do fisioterapeuta na *PhysioApp*.

Premindo o botão número 4 da Figura 2.8, é exibido o menu com um calendário da Figura 2.11, onde pode ser definido a data de nascimento do fisioterapeuta. Clicando no ano este pode ser alterado. Por outro lado, ao clicar nas setas horizontais é possível alterar o mês de nascimento. Por último, clicando no dia do mês e no botão OK, a nova data de nascimento é adicionada ao formulário.

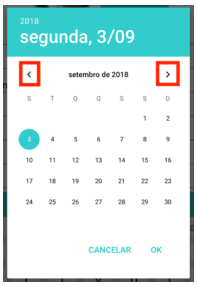


Figura 2.11 – Menu da data de nascimento do fisioterapeuta na *PhysioApp*.

Finalmente, clicando no botão número 5 da Figura 2.8 o pedido de criação de um novo fisioterapeuta é enviado. Caso não exista nesse momento nenhum fisioterapeuta no sistema com o mesmo *e-mail* ou *username* é exibida a mensagem da Figura 2.12, referindo o sucesso da operação.

Figura 2.12 – Mensagem de sucesso na criação do fisioterapeuta na *PhysioApp*.

Por outro lado, caso algum dos dados referidos esteja em uso por outro fisioterapeuta, será mostrada uma das três mensagens apresentadas na Figura 2.13. Sendo que para registar o fisioterapeuta deverão ser alterados os campos repetidos.

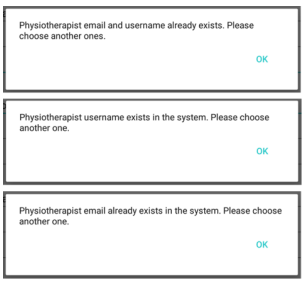


Figura 2.13 – Mensagem de erro na criação do fisioterapeuta na *PhysioApp*.

2.2.4. Selecionar fisioterapeuta

Clicando num elemento da lista de fisioterapeutas e no botão de *OK*, marcado com o número 4 na Figura 2.5, o administrador poderá aceder à página de perfil do fisioterapeuta em causa, como apresentado na Figura 2.14.

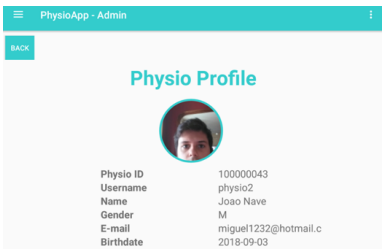


Figura 2.14 – Visualização do perfil de um fisioterapeuta pelo administrador na *PhysioApp*.

Caso prima o referido botão de *OK* sem selecionar nenhum paciente, será exibida a mensagem da Figura 2.15, que indica que deve ser selecionado um fisioterapeuta.

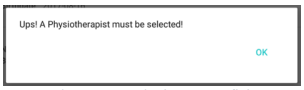


Figura 2.15 – Mensagem de erro ao selecionar um fisioterapeuta na *PhysioApp*.

2.3. Manual do fisioterapeuta

Quando o fisioterapeuta entra na *PhysioApp*, este é encaminhado para a sua página de perfil (ver Figura 2.16).

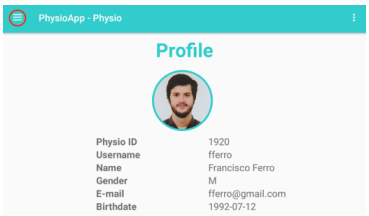


Figura 2.16 – Perfil do fisioterapeuta na *PhysioApp*.

No seu perfil, o fisioterapeuta pode visualizar as suas informações pessoais. como o seu *username*, nome, género, *e-mail* ou dia de aniversário. Clicando no botão marcado a vermelho na Figura 2.16, este acede ao menu lateral exibido na Figura 2.17.

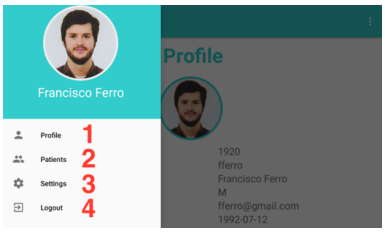


Figura 2.17 – Menu lateral do fisioterapeuta na *PhysioApp*.

Neste o menu lateral, o fisioterapeuta tem à sua disposição os seguintes 4 botões marcados com números de 1 a 4:

- **1** → Encaminha o fisioterapeuta de novo para o seu perfil.
- **2** → Mostra a página dos pacientes.
- **3** → Mostra a página das definições.
- **4** → Realiza o logout do fisioterapeuta na aplicação.

Devido à maior complexidade das acções levadas a cabo a partir dos botões 2 e 3 da Figura 2.17, estes são mais detalhados nas secções 2.3.1 e 2.3.2, respectivamente.

2.3.1. Pacientes

Clicando no botão *Patients*, marcado com o número 2 na Figura 2.17, o paciente é deslocado para a página apresentada na Figura 2.18.

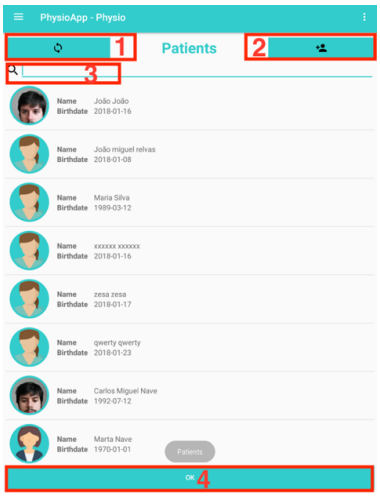


Figura 2.18 – Página dos pacientes na *PhysioApp*.

Nesta página, o fisioterapeuta pode realizar as seguintes acções:

- **Sincronizar os pacientes:** Clicando no botão 1 da Figura 2.18, o fisioterapeuta realiza a sincronização dos pacientes com o servidor. Caso haja algum paciente novo ser-lhe-á apresentada a mensagem do topo da Figura 2.19, caso contrário será exibida a mensagem da base da mesma figura.

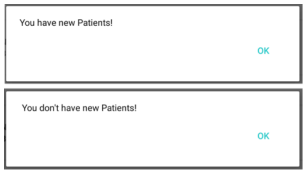


Figura 2.19 – Mensagens da sincronia dos pacientes na *PhysioApp*.

- **Criar pacientes:** Clicando no botão 2 da Figura 2.18, é apresentada a página de criação de pacientes, com um formulário que deverá ser preenchido com as informações do paciente (ver Figura 2.20). Se pretender regressar à página dos fisioterapeutas deverá clicar no botão marcado com o número 1 na Figura 2.18.

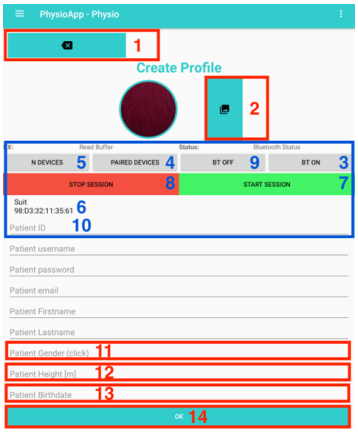


Figura 2.20 – Página de criação do paciente na *PhysioApp*.

Clicando no botão número 2 da Figura 2.18, é mostrado o menu para adicionar uma fotografia mostrado na Figura 2.21. Nesse menu, o fisioterapeuta pode tirar ou escolher na galeria do dispositivo uma fotografia para identificar o paciente.

Figura 2.21 – Menu da fotografia do paciente na *PhysioApp*.

Os botões ou campos identificados na Figura 2.20 com os números entre 3 e 9, fazem parte do menu *Bluetooth* que é usado para definir o registar o número do cartão do paciente. O dispositivo recebe o número do cartão do paciente quando a comunicação é establecida com o *PhysioRegister*, o módulo leitor de cartões *RFID,* e o cartão é lido por esse mesmo leitor. Os botões do menu *Bluetooth* devem ser acionados sequencialmente e em ordem crescente segundo o seu número na figura, realizando as seguintes acções:

- **3 →** Este botão liga o *Bluetooth* do dispositivo.
- **4 →** Este botão lista no campo número 7 os dispositivos emparelhados com o dispositivo móvel utilizado. Caso o módulo leitor de cartões não esteja ainda emparelhado deve ser usado o botão número 5 para esse efeito.
- **5 →** Este botão lista no campo número 7 os dispositivos com o *Bluetooth* ligado que estão disponíveis para emparelhar.
- **6 →** Esta lista apresenta, dependendo se se premiu o botão número 4 ou 5, os dispositivos emparelhados, ou os dispositivos disponíveis para emparelhar. O fisioterapeuta deve pesquisar na lista e selecionar o dispositivo com o nome *physioregister*.
- **7 →** Este botão inicia a comunicação *Bluetooth* com o módulo *PhysioRegister*. Esta comunicação deve ser realizada depois da leitura do cartão por parte do módulo referido.
- **8 →** Este botão termina a comunicação *Bluetooth* com o módulo *PhysioRegister*. No entanto só deverá ser usado quando verificado que o número do cartão lido já se encontra no campo 10.
- **9 →** Se desejar, o fisioterapeuta poderá desligar o *Bluetooth* do dispositivo ao clicar neste botão.

Por outro lado, ao premir o botão número 11 da Figura 2.20, é mostrado ao fisioterapeuta um menu de selecção do sexo do paciente. Existem duas opções, *Male* ou *Female*, e só poderá ser escolhido uma das duas. Premindo numa das opções, o género do paciente ficará identificado (ver Figura 2.22).

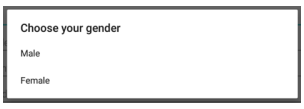


Figura 2.22 – Menu de selecção do género do paciente na *PhysioApp*.

Premindo o campo 12, é exibido o menu de selecção da altura do paciente da Figura 2.23. Depois de selecionada a altura, em metros, esta irá aparecer identificada no formulário da Figura 2.20.

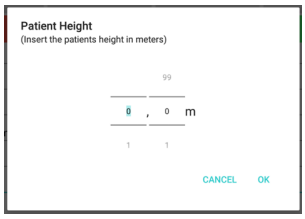


Figura 2.23 – Menu de selecção da altura do paciente na *PhysioApp*.

Ao carregar no campo 13 da Figura 2.20, é exibido o menu de seleção da data de nascimento do paciente, com um calendário, apresentado na Figura 2.24. Após a data ser selecionada, esta irá aparecer identificada no formulário da Figura 2.20.

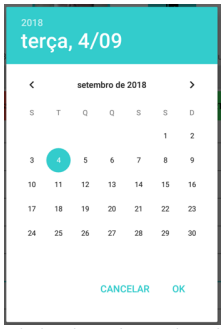


Figura 2.24 – Menu da data de nascimento do paciente na *PhysioApp*.

Finalmente, clicando no botão número 14 da Figura 2.20 o pedido de criação de um novo paciente é enviado. Caso não exista nesse momento nenhum paciente no sistema com o mesmo *e-mail*, *username* ou número de cartão, é exibida a mensagem da Figura 2.25, referindo o sucesso da operação.

Figura 2.25 – Mensagem de sucesso na criação do paciente na *PhysioApp*.

Por outro lado, caso algum dos dados referidos já esteja em uso por outro paciente, será mostrada uma das mensagens apresentadas na Figura 2.26. Sendo que para registar o fisioterapeuta deverão ser alterados os campos repetidos.

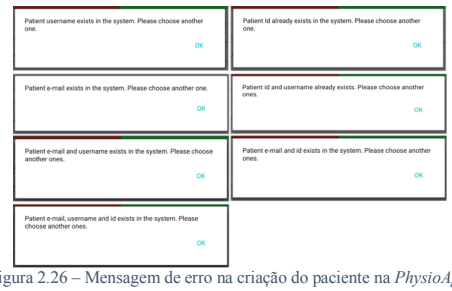


Figura 2.26 – Mensagem de erro na criação do paciente na *PhysioApp*.

- **Pesquisa por pacientes:** O fisioterapeuta tem a possibilidade de pesquisar um paciente presente na lista. Para isso deve inserir no campo de texto marcado na Figura 2.18, com o número 3, alguma informação identificativa do paciente em causa. Estes poderão ser encontrados pelo seu nome ou o seu *e-mail* (ver Figura 2.27).

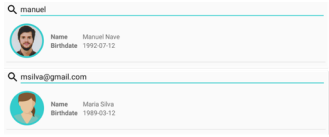


Figura 2.27 – Pesquisa por pacientes na *PhysioApp*.

- **Selecionar paciente:** Clicando num elemento da lista de pacientes e no botão de *OK*, marcado com o número 4 na Figura 2.18, o fisioterapeuta poderá aceder à página de perfil do paciente em causa, como apresentado na Figura 2.28. Sendo que se pretender regressar à página com a lista dos pacientes, basta premir o botão marcado na Figura 2.28 com o número 1.

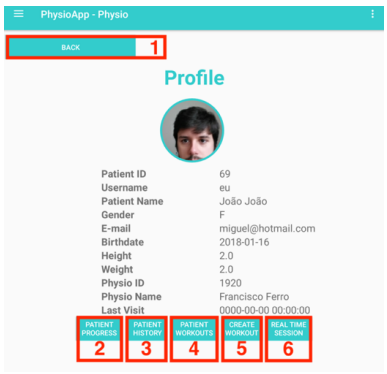


Figura 2.28 – Visualização do perfil de um paciente na *PhysioApp*.

No perfil do paciente existem os seguintes botões numerados de 2 a 6 que permitem ao fisioterapeuta realizar diferentes acções:

2 → Este botão encaminha o fisioterapeuta para uma página com progresso do paciente. Isto é, uma página com informações estatísticas das últimas 5 sessões realizadas pelo paciente selecionado (ver Figura 2.29).

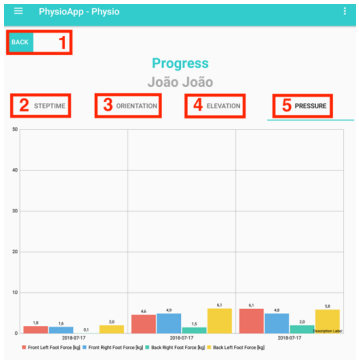


Figura 2.29 – Página do progresso de um paciente do fisioterapeuta na *PhysioApp*.

Nesta página existem 4 botões, numerados de 2 a 5, que apresentam gráficos com valores médios das seguintes métricas das últimas 5 sessões: Tempo por passo (2); ângulos de orientação (3); elevação bilateral (4); força nos pés do andarilho (5).

3 → Este botão encaminha o fisioterapeuta para a página com o histórico de sessões do paciente, exibida na Figura 2.30.

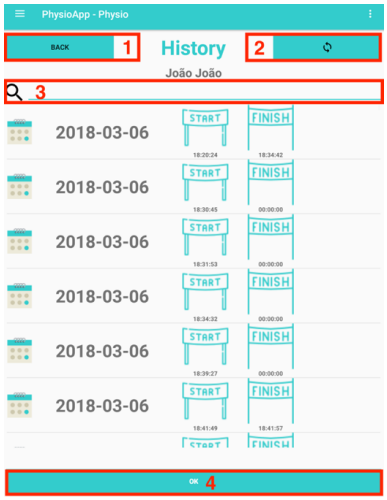


Figura 2.30 – Histórico de sessões de um paciente do fisioterapeuta na *PhysioApp*.

Nesta página o fisioterapeuta pode realizar as seguintes 4 acções:

- **Regresso ao perfil do paciente:** Clicando no botão marcado com o número 1 na Figura 2.30.
- **Sincronia das sessões de fisioterapia:** Clicando no botão marcado com o número 2 da Figura 2.30, o fisioterapia realiza a sincronização das sessões com o servidor. Caso haja alguma sessão nova ser-lhe-á apresentada a mensagem do topo da Figura 2.31, caso contrário será exibida a mensagem da base da mesma figura.

Figura 2.31 – Mesagens de sincronia das sessões de um paciente do fisioterapeuta na *PhysioApp*.

- **Pesquisa por uma sessão de fisioterapia:** Inserindo no campo de texto marcado na Figura 2.30 com o número 3, uma data parcial ou completa, o fisioterapeuta poderá pesquisar uma sessão (ver Figura 2.32).

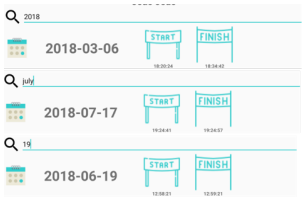


Figura 2.32 – Pesquisa por sessões de fisioterapia de um paciente na *PhysioApp*.

- **Visualização de uma sessão de fisioterapia:** Selecionando uma sessão presente na lista da Figura 2.30 e clicando no botão de *OK* marcado na com o número 4 na mesma figura, o fisioterapeuta pode aceder aos dados da sessão selecionada. Caso clique somente no botão de *OK*, será mostrada a mensagem de erro da Figura 2.33.

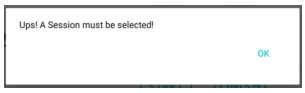


Figura 2.33 – Mensagem de erro quando o fisioterapeuta seleciona uma sessão na *PhysioApp*.

Caso o fisioterapeuta selecione a sessão de forma correta, será direcionado à página dessa sessão, exibida na Figura 2.34.

Figura 2.34 – Página de sessão de fisioterapia de um paciente na *PhysioApp*.

Na página de uma sessão de fisioterapia, como a apresentada na Figura 2.34, o fisioterapeuta pode regressar à lista de sessões (botão 1) ou visualizar os dados relativos à sessão selecionada. Os dados mostrados dividem-se nas seguintes 3 categorias:

- **Dados temporais:** Número de sessão (2); data da sessão (3); hora de início da sessão (4); hora do fim da sessão (5).
- **Métricas calculadas:** Altura média e máxima do lado esquerdo (6); número de passos (7); altura média e máxima do lado direito (8); área do equilíbrio (9); perímetro do equilíbrio (10); ponto de equilíbrio médio do paciente (11).
- **Gráficos da sessão:** Gráfico de orientação (12); gráfico de elevação (13); gráfico de equilíbrio do paciente (14); gráfico de forças nos pés do andarilho (15).

4 → Este botão encaminha o fisioterapeuta para a página com o histórico de planos de exercício do paciente, exibida na Figura 2.35.

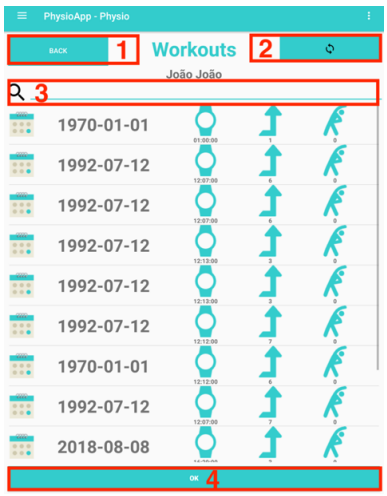


Figura 2.35 – Histórico dos planos de exercício de um paciente na *PhysioApp*.

Nesta página, o fisioterapeuta pode realizar as seguintes acções:

- **Voltar para a página do paciente selecionado:** Clicando no botão 1 da Figura 2.35, o fisioterapeuta regressa à página do perfil do paciente previamente selecionado.
- **Sincronizar os planos de exercício físico:** Clicando no botão 2 da Figura 2.35, o fisioterapeuta realiza a sincronização dos planos de exercício físico com o servidor. Caso haja algum novo plano, ser-lhe-á apresentada a mensagem do topo da Figura 2.36, caso contrário será exibida a mensagem da base da mesma figura.

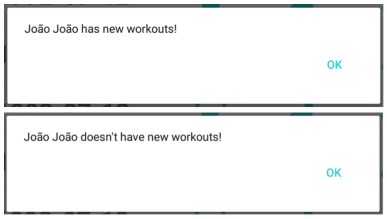


Figura 2.36 – Mensagens de sincronia dos planos de exercício de um paciente na *PhysioApp*.

- **Pesquisar um plano de exercício físico:** O fisioterapeuta pode pesquisar por uma sessão de um paciente em específico inserindo no campo de texto com o número 3 na Figura 2.35, dados identificativos da mesma. Os dados reconhecidos são, a data completa ou parcial do plano (ver Figura 2.37).

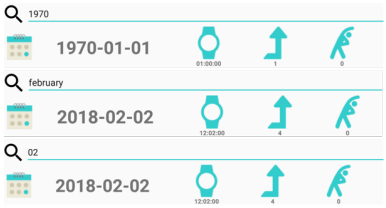


Figura 2.37 – Exemplos de pesquisa do fisioterapeuta por um plano de exercício na *PhysioApp*.

- **Selecionar um plano de exercício físico:** Caso deseje selecionar um plano de exercício para ver os dados do mesmo, o fisioterapeuta deve clicar sobre um plano da lista e só depois clicar no botão *OK*, marcado com o número 4 na Figura 2.35. Se assim não o fizer, a mensagem da Figura 2.38 será apresentada.

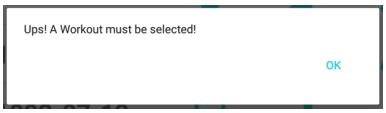


Figura 2.38 – Mensagem de erro quando o fisioterapeuta seleciona uma sessão na *PhysioApp*.

Caso o fisioterapeuta selecione o plano de forma correta, será direcionado à página do plano selecionado, exibida na Figura 2.39.

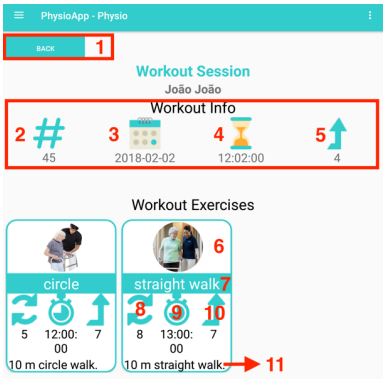


Figura 2.39 – Página de plano de exercício de um paciente na *PhysioApp*.

Na página de um plano de exercícios, como a apresentada na Figura 2.39, o fisioterapeuta pode regressar à lista de planos (botão 1) ou visualizar os dados relativos ao plano selecionado. Os dados mostrados dividem-se nas 2 seguintes categorias:

- **Dados do plano:** Número do plano (2); data do plano (3); hora de início do plano (4); hora do fim do plano (5).

- **Dados dos exercícios:** Fotografia do exercício (6); título do exercício (7); número de repetições (8); duração do exercício (9); nível de dificuldade do exercício (10); descrição do exercício (11).

5 → Este botão encaminha o fisioterapeuta para a página de criação de um plano de exercícios, exibida na Figura 2.40.

Figura 2.40 – Página de criação de um plano de exercícios na *PhysioApp*.

Nesta página o fisioterapeuta pode regressar aos planos de exercício do paciente, clicando no botão *back* (marcado com o número 1 na Figura 2.40) ou criar um novo plano de exercícios preenchendo e submetendo o formulário presente na mesma figura.

O formulário da criação do plano de exercícios divide-se nas seguintes duas categorias:

- **Campos do plano de exercícios:** Estes são o campo da data (2); da duração (3); e da dificuldade do plano de exercício físico a criar (4).

Quando o fisioterapeuta pressiona o campo número 2 (ver Figura 2.40), é-lhe apresentado o menu da Figura 2.41, com um calendário para selecionar a data do plano a criar.

Figura 2.41 – Menu da escolha da data do plano de exercício na *PhysioApp*.

Por outro lado, quando o fisioterapeuta clica no campo número 3 (ver Figura 2.40), é-lhe apresentado o menu da Figura 2.42 com um relógio para selecionar a hora do plano a criar.

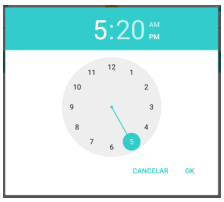


Figura 2.42 – Menu da escolha da hora do plano de exercício na *PhysioApp*.

Finalmente, quando o campo número 4 é clicado (ver Figura 2.40), é exibido o menu da Figura 2.43, com um nível para escolher o grau de dificuldade do plano de exercícios a ser criado.

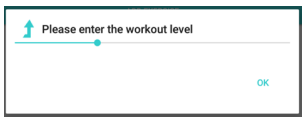


Figura 2.43 – Menu da escolha do nível de dificuldade do plano de exercício na *PhysioApp*.

- **Campos de um exercício:** Estes são o campo da fotografia ilustrativa do exercício (6); o do título do exercício (7); o do número de repetições do exercício (8); o da duração do exercício (9); o da dificuldade do exercício (10); e o da descrição do exercício (11).

Quando o fisioterapeuta pressiona o botão com o número 6 (ver Figura 2.40), é-lhe apresentado o menu da Figura 2.44, onde este pode escolher

tirar ou selecionar uma fotografia da galeria do dispositivo para ilustrar o exercício a ser criado.

Figura 2.44 – Menu da escolha da fotografia do exercício a criar na *PhysioApp*.

Ao clicar no campo com o número 7 (ver Figura 2.40), o fisioterapeuta pode escrever o título do exercício que está a ser criado. Por outo lado, ao premir o campo com o número 8 (ver Figura 2.40), é apresentada ao fisioterapeuta o menu da Figura 2.45 para selecionar o número de repetições do exercício que está a ser criado.

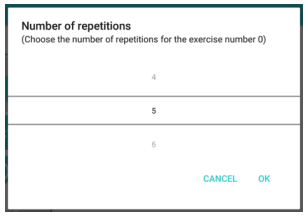


Figura 2.45 – Menu da escolha do número de repetições do exercício na *PhysioApp*.

Ao ser selecionado o campo com o número 9 (ver Figura 2.40), é exibida a janela da Figura 2.46, com um relógio, onde pode ser selecionada a duração do exercício que está a ser criado.

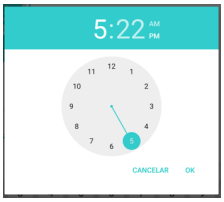


Figura 2.46 – Menu da escolha da duração do exercício na *PhysioApp*.

Quando o campo marcado com o número 10 na Figura 2.40 é clicado, é exibido o menu da Figura 2.47 com um nível para escolher o grau de dificuldade do exercício que está a ser criado.

Figura 2.47 – Menu da escolha do nível de dificuldade do exercício na *PhysioApp*.

Finalmente, quando o campo com o número 11 é clicado (ver Figura 2.40), é apresentado o menu da Figura 2.48, com um campo de texto onde é introduzida uma breve descrição do exercício que está a ser criado.

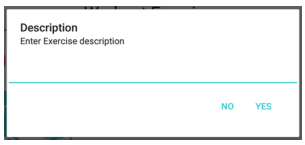


Figura 2.48 – Menu da escolha da descrição do exercício na *PhysioApp*.

Adicionalmente, o botão marcado com o número 5 na Figura 2.40 pode ser usado para adicionar mais exercícios ao plano que está a ser criado.

Finalmente, o botão marcado com o número 12 na Figura 2.40, é usado para submeter o pedido de criação de um novo plano de exercício. Caso o mesmo seja bem-sucedido é mostrada a mensagem da Figura 2.49.

Figura 2.49 – Mensagem de sucesso na criação de um novo plano de exercício na *PhysioApp*.

Caso o fisioterapeuta se tenha esquecido de preencher algum campo de todo o formulário da criação do plano, é exibida a mensagem de erro da Figura 2.50.

Figura 2.50 – Mensagem de erro na criação de um novo plano de exercício na *PhysioApp*.

6 → Este botão encaminha o fisioterapeuta para a página de realização de uma sessão de fisioterapia em tempo real, apresentada na Figura 2.51.

Figura 2.51 – Página da sessão em tempo real da *PhysioApp*.

Nesta página, apresentada na Figura 2.51, o fisioterapeuta pode regressar ao perfil do paciente previamente selecionado (botão 1) ou começar uma sessão de fisioterapia em tempo real. São apresentados nesta página os seguintes menus:

- **Menu de comunicação *Bluetooth*:** Este menu permite estabelecer a comunicação *Bluetooth* do dispositivo móvel com o andarilho, sendo absolutamente indepesável a sua utilização para a realização de uma sessão com a visualização de dados em tempo real. O menu contém os seguintes elementos, assinalados com os números de 2 a 8 na Figura 2.51, que devem ser usados sequencialmente:
  - **2→** Este botão deve ser premido para ligar o *Bluetooth* do dispositivo móvel, caso este ainda não esteja ligado.
  - **3→** Este botão realiza a listagem dos dispositivos emparelhados no elemento marcado com o número 5 na Figura 2.51. Este botão só deverá ser premido caso o andarilho já esteja emparelhado. Caso contrário deve ser usado o botão marcado com o número 4 na mesma figura.
  - **4→** Este botão realiza a listagem dos dispositivos não emparelhados e disponíveis no elemento marcado com o número 5 na Figura 2.51. Este botão deverá ser premido caso o dispositivo ainda não esteja emparelhado com o andarilho. Caso contrário

deverá ser utilizado o botão marcado com o número 3 na mesma figura.

- **5→** Este elemento trata-se de uma lista de dispositivos *Bluetooth* disponíveis. O fisioterapeuta deverá pesquisar e premir o dispositivo identificado pelo nome *smartwalker* nessa mesma lista para estabelecer a comunicação *Bluetooth* com o andarilho.
- **6→** Depois de ter sido estabelecida a comunicação *Bluetooth* com o andarilho, o fisioterapeuta deverá clicar sobre este botão para iniciar a sessão de fisioterapia.
- **7→** Após o início da sessão de fisioterapia, e quando for desejado terminá-la, o fisioterapeuta deverá clicar neste botão.
- **8 →** Adicionalmente, o fisioterapeuta poderá desligar o *Bluetooth* do dispositivo carregando neste botão.

- **Menu dos gráficos:** Este menu apresenta os seguintes cinco botões (9 a 13) que podem ser selecionados para ver as métricas calculadas em tempo real:
  - **9 →** Este botão permite ver o gráfico relativo à orientação do paciente. Isto é, os seus ângulos de orientação (*Roll*, *Pitch* e *Yaw*) durante a sessão de fisioterapia.
  - **10 →** Este botão permite ver o gráfico relativo à elevação do andarilho. Isto é, a elevação bilateral do andarilho por parte do paciente em função do tempo, ao longo da sessão de fisioterapia.
  - **11 →** Este botão permite ver o gráfico relativo à elevação bilateral instantânea do andarilho por parte do paciente ao longo do tempo da sessão de fisioterapia.
  - **12 →** Este botão permite ver o gráfico relativo ao equilíbrio do paciente. Isto é, os pontos de equilíbrio instantâneo do paciente ao longo do tempo da sessão de fisioterapia.
  - **13 →** Este botão permite ver o gráfico relativo às forças exercidas sobre cada pé do andarilho, por parte do paciente, ao longo da sessão de fisioterapia.

2.3.2. Definições

Clicando no botão *Settings*, marcado com o número 3 na Figura 2.17, o fisioterapeuta é deslocado para a página apresentada na Figura 2.52.

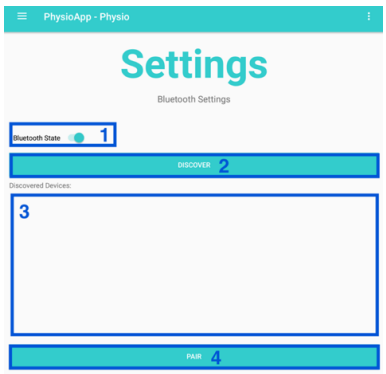


Figura 2.52 – Menu de definições do *Bluetooth* na *PhysioApp*.

Nesta página o fisioterapeuta pode realizar o emparelhamento do dispositivo móvel com o protótipo de andarilho desenvolvido. Para isso deverá realizar as seguintes 4 acções sequencialmente:

- **Ligar o *Bluetooth* do dispositivo móvel:** Se o *Bluetooth* do dispositivo não estiver ligado, o fisioterapeuta deve usar o botão marcado com o número 1 na Figura 2.52 para o ligar.
- **Começar a descoberta de dispositivos *Bluetooth*:** Para realizar a descoberta de dispositivos *Bluetooth*, o fisioterapeuta deve clicar no botão de descoberta, marcado com o número 2 na Figura 2.52. Os dispositivos encontrados aparecem na lista assinalada com o número 3 na mesma figura.
- **Selecionar o andarilho:** O fisioterapeuta deve pesquisar e selecionar o dispositivo identificado com o nome *smartwalker*, presente na lista marcada com o número 3 na Figura 2.52.
- **Emparelhar o dispositivo com o andarilho:** Depois de selecionar o andarilho na lista, o fisioterapeuta deverá clicar no botão presente na Figura 2.52 com o número 4, de maneira a realizar o emparelhamento *Bluetooth* do dispositivo móvel com o andarilho.

2.4. Manual do paciente

Quando o paciente entra na *PhysioApp,* este é encaminhado para a sua página de perfil (ver Figura 2.53).

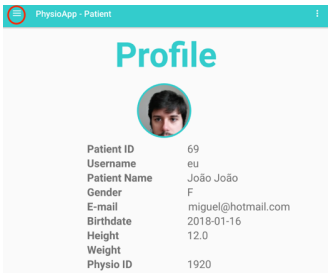


Figura 2.53 – Perfil do paciente na *PhysioApp*.

No seu perfil, o paciente pode visualizar as suas informações pessoais. como o seu *username*, nome, género, *e-mail* ou dia de aniversário. Clicando no botão marcado a vermelho (ver Figura 2.53), este acede ao menu lateral exibido na Figura 2.54.

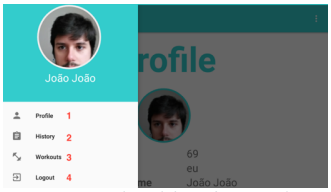


Figura 2.54 – Menu lateral do paciente na *PhysioApp*.

Neste o menu lateral, o paciente tem à sua disposição os seguintes 4 botões marcados com números de 1 a 4:

- **1** → Encaminha o paciente de novo para o seu perfil.
- **2** → Mostra o histórico de sessões de fisioterapia do paciente.
- **3** → Mostra o histórico de planos de exercício físico do paciente.
- **4** → Realiza o *logout* do paciente na aplicação.

Devido à maior complexidade das acções levadas a cabo a partir dos botões 2 e 3 da Figura 2.54, estes são mais detalhados nas secções 2.4.1 e 2.4.2, respectivamente.

2.4.1. Histórico de sessões

Clicando no botão *History*, marcado com o número 2 na Figura 2.54, o paciente é deslocado para a página apresentada na Figura 2.55.

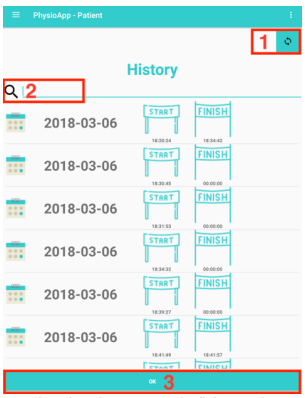


Figura 2.55 – Histórico das sessões de fisioterapia na *PhysioApp*.

Nesta página, o paciente pode realizar as seguintes acções:

- **Sincronizar as sessões de fisioterapia:** Clicando no botão 1 da Figura 2.55, o paciente realiza a sincronização das sessões com o servidor. Caso haja alguma sessão nova ser-lhe-á apresentada a mensagem do topo da Figura 2.56, caso contrário será exibida a mensagem da base da mesma figura.

Figura 2.56 – Mensagens de sincronia das sessões do paciente na *PhysioApp*.

- **Pesquisar uma sessão de fisioterapia:** O paciente pode pesquisar por uma sessão em específico inserindo no campo de texto com o número 2 na Figura 2.55 dados identificativos da mesma. Os dados reconhecidos são, a data por inteiro ou parcial da sessão (ver Figura 2.57).

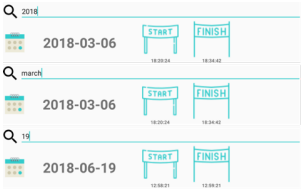


Figura 2.57 – Exemplos de pesquisa de sessões de fisioterapia na *PhysioApp*.

- **Selecionar uma sessão de fisioterapia:** Caso deseje selecionar uma sessão de fisioterapia para ver os dados da mesma, o paciente deve clicar sobre uma sessão

da lista e só depois clicar no botão *OK*, marcado com o número 3 na Figura 2.55. Se assim não o fizer, a mensagem da Figura 2.58 será apresentada.

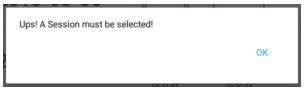


Figura 2.58 – Mensagem de erro quando o paciente seleciona uma sessão na *PhysioApp*.

Caso o paciente selecione a sessão de forma correta, será direcionado à página da sessão selecionada, exibida na Figura 2.59.

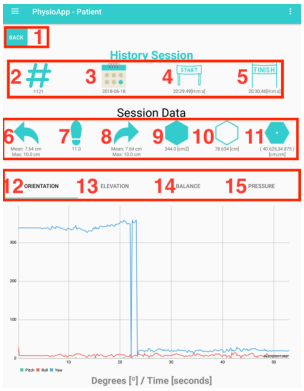


Figura 2.59 – Página de sessão de fisioterapia do paciente na *PhysioApp*.

Na página de uma sessão de fisioterapia, como a apresentada na Figura 2.59, o paciente pode regressar à lista de sessões de fisioterapia (botão 1) ou visualizar os dados relativos à sessão selecionada. Os dados mostrados dividem-se nas seguintes 3 categorias:

- **Dados temporais:** Número de sessão (2); data da sessão (3); hora de início da sessão (4); hora do fim da sessão (5).
- **Métricas calculadas:** Altura média e máxima do lado esquerdo (6); número de passos (7); altura média e máxima do lado direito (8); área do equilíbrio (9); perímetro do equilíbrio (10); ponto de equilíbrio médio do paciente (11).
- **Gráficos da sessão:** Gráfico de orientação (12); gráfico de elevação (13); gráfico de equilíbrio do paciente (14); gráfico de forças nos pés do andarilho (15).

2.4.2. Histórico de planos de exercício físico

Clicando no botão *Workouts*, marcado com o número 3 na Figura 2.54, o paciente é deslocado para a página apresentada na Figura 2.60.

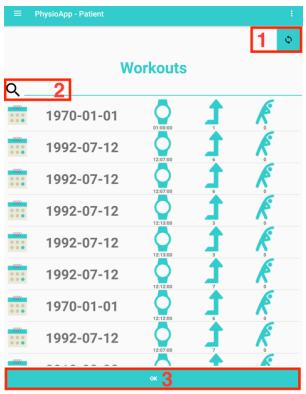


Figura 2.60 – Histórico dos planos de exercício do paciente na *PhysioApp*.

Nesta página, o paciente pode realizar as seguintes acções:

- **Sincronizar os planos de exercício físico:** Clicando no botão 1 da Figura 2.60, o paciente realiza a sincronização dos planos de exercício físico com o servidor. Caso haja algum novo plano, ser-lhe-á apresentada a mensagem do topo da Figura 2.61, caso contrário será exibida a mensagem da base da mesma figura.

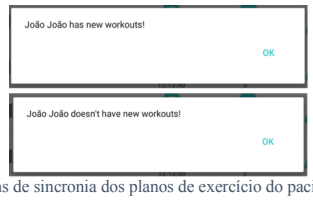


Figura 2.61 – Mensagens de sincronia dos planos de exercício do paciente na *PhysioApp*.

- **Pesquisar um plano de exercício físico:** O paciente pode pesquisar por um plano em específico inserindo no campo de texto com o número 2 na Figura 2.60 dados identificativos do mesmo. Os dados reconhecidos são, a data completa ou parcial do plano (ver Figura 2.62).

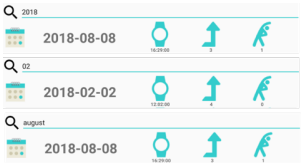


Figura 2.62 – Exemplos de pesquisa do paciente por um plano de exercício na *PhysioApp*.

- **Selecionar um plano de exercício físico:** Caso deseje selecionar um plano de exercício para ver os dados do mesmo, o paciente deve clicar sobre um plano da lista e só depois clicar no botão *OK*, marcado com o número 3 na Figura 2.60. Se assim não o fizer, a mensagem da Figura 2.63 será apresentada.

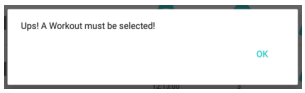


Figura 2.63 – Mensagem de erro quando o paciente seleciona um plano de exercício na *PhysioApp*.

Caso o paciente selecione o plano de forma correta, será direcionado à página do plano selecionado, exibida na Figura 2.64.

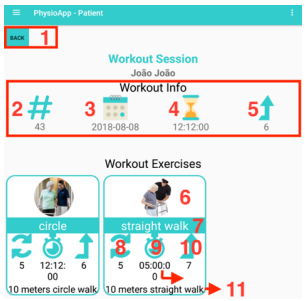


Figura 2.64 – Página de plano de exercício do paciente na *PhysioApp*.

Na página de um plano de exercícios, como a apresentada na Figura 2.64, o paciente pode regressar à lista de planos (botão 1) ou visualizar os dados relativos ao plano selecionado. Os dados mostrados dividem-se nas seguintes 2 categorias:

- **Dados do plano:** Número do plano (2); data do plano (3); hora de início do plano (4); hora do fim do plano (5).
- **Dados dos exercícios:** Fotografia do exercício (6); título do exercício (7); número de repetições (8); duração do exercício (9); nível de dificuldade do exercício (10); descrição do exercício (11).

Capítulo 3 – *PhysioWebapp*

Neste capítulo apresenta-se a descrição e funcionamento da aplicação *web* do sistema *IoPhyR,* a *PhysioWebapp*.

3.1. Descrição da aplicação *web*

A aplicação *web* desenvolvida e pertencente ao sistema *IoPhyR*, a *PhysioWebapp*, foi desenvolvida utilizando a *Framework material design bootstrap* de maneira a correr em todos os dispositivos independentemente do tamanho ou resolução do seu ecrã. O desenvolvimento desta plataforma teve como objectivo tornar o sistema disponível para os utilizadores que não possuam um dispositivo móvel com o sistema operativo *Android* (requisito da aplicação móvel desenvolvida, a *PhysioApp*).

O utilizador que pretenda utilizar a *PhysioWebapp* deve estar previamente registado no sistema. Podendo estar-lhe atribuído o perfil de administrador, fisioterapeuta ou paciente. Ao entrar na aplicação *web* o utilizador depara-se com a página inicial de login, onde deve inserir as suas credenciais, isto é, o seu *username* e *password* (ver Figura 3.1).

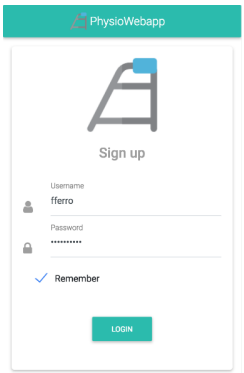


Figura 3.1 – Página de entrada da *PhysioWebapp*.

Caso o login seja bem-sucedido, será apresentada a mensagem do topo da Figura 3.2, caso contrário será mostrada a da base da mesma figura.

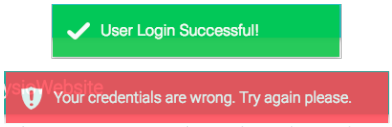


Figura 3.2 – Mensagens de entrada na *PhysioWebapp*.

Devido às particularidades de casa tipo de utilizador são apresentados 3 manuais, um para cada um deles.

3.2. Manual do administrador

Quando o administrador do sistema entra na *PhysioWebapp*, este é encaminhado para a sua página de perfil (ver Figura 3.3).

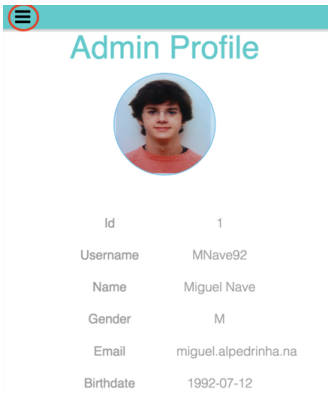


Figura 3.3 – Perfil do administrador na *PhysioWebapp*.

No perfil pode visualizar as suas informações pessoais, podendo também clicar no botão marcado a vermelho na Figura 3.3 para visualizar o menu lateral da Figura 3.4.

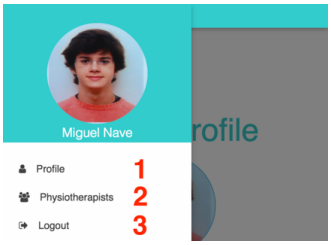


Figura 3.4 – Menu lateral do administrador na *PhysioWebapp*.

Neste menu lateral, o administrador tem à sua disposição os seguintes 4 botões marcados com números de 1 a 3:

- **1→** Redireciona o administrador novamente para o seu perfil.
- **2→** Encaminha o administrador para a área dos fisioterapeutas.
- **3→** Realiza o *logout* do administrador.

Carregando no botão número 2 da Figura 3.4, o administrador é redirecionado para a área dos fisioterapeutas. Lá pode visualizar a lista de fisioterapeutas do sistema, sincronizá-la ou criar ou selecionar um fisioterapeuta. A primeira acção é executada quando o administrador clica no botão número 1 da Figura 3.5. As restantes acções são descritas ao pormenor nas secções 3.2.1 e 3.2.2.

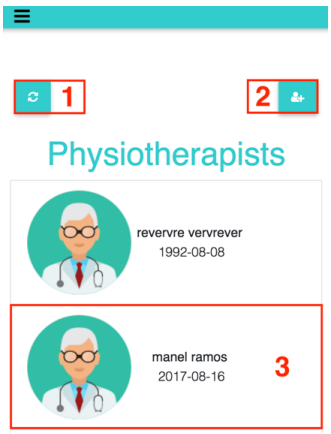


Figura 3.5 – Página dos fisioterapeutas na *PhysioWebapp*.

3.2.1. Criar fisioterapeutas

Ao clicar no botão número 2 da Figura 3.5, o administrador acede à página da Figura 3.6. Nesta página, é apresentado um formulário onde inserindo os dados requeridos, é possível realizar o pedido de criação dum novo fisioterapeuta. Se desejar, o administrador pode regressar à lista de fisioterapeutas clicando no botão 1 da Figura 3.6.

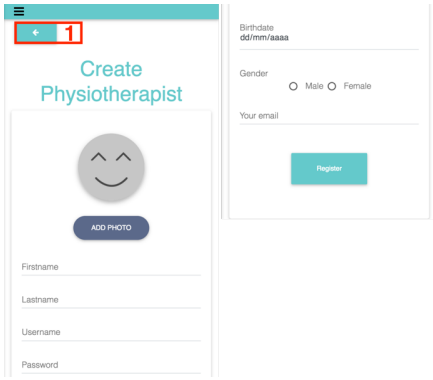


Figura 3.6 – Página de criação de fisioterapeutas na *PhysioWebapp*.

Para o pedido ser bem-sucedido, o *username* e *e-mail* escolhidos não devem existir no sistema. Caso não existam será mostrada a mensagem de sucesso da Figura 3.7.

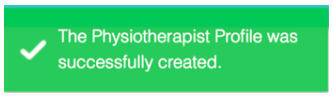


Figura 3.7 – Mensagem de sucesso na criação de um fisioterapeuta na *PhysioWebapp*.

Caso contrário, será mostrada uma das mensagens presentes na Figura 3.8, e o fisioterapeuta não será criado.

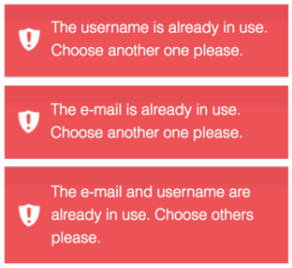


Figura 3.8 – Mensagem de erro na criação de um fisioterapeuta na *PhysioWebapp*.

3.2.2. Selecionar fisioterapeuta

Clicando num elemento da lista de fisioterapeutas da Figura 3.5, como o marcado com o número 3, o administrador é direcionado para o perfil do fisioterapeuta selecionado. Nesse perfil, apresentando na Figura 3.9, o administrador pode ver os dados pessoais do fisioterapeuta ou regressar à lista de fisioterapeutas clicando no botão marcado com o número 1.

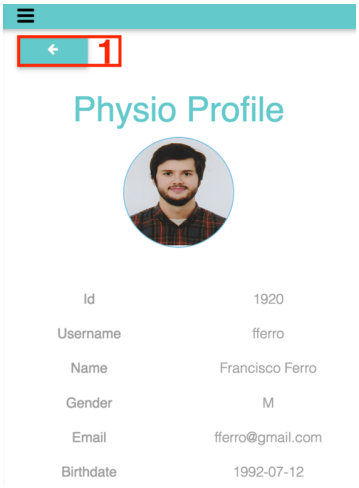


Figura 3.9 – Visualização do perfil de um fisioterapeuta por parte do administrador na *PhysioWebapp*.

3.3. Manual do fisioterapeuta

Quando um fisioterapeuta entra na *PhysioWebapp*, este é encaminhado para a sua página de perfil (ver Figura 3.10).

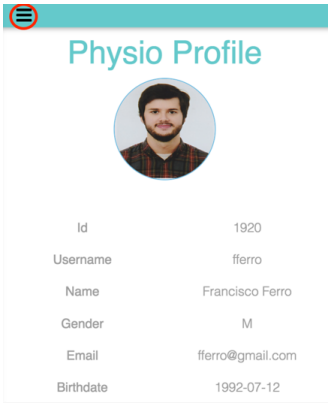


Figura 3.10 – Página de perfil do fisioterapeuta na *PhysioWebapp*.

No seu perfil, o fisioterapeuta pode visualizar as suas informações pessoais, assim como clicar no botão marcado a vermelho na Figura 3.10 de modo a aceder ao menu lateral exibido na Figura 3.11.

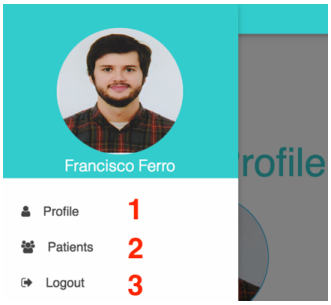


Figura 3.11 – Menu lateral do fisioterapeuta na *PhysioWebapp*.

Neste menu lateral, o fisioterapeuta tem à sua disposição os seguintes 3 botões marcados com números de 1 a 3:

- **1** → Encaminha o fisioterapeuta de novo para o seu perfil.
- **2** → Direciona o fisioterapeuta para a área dos seus pacientes.
- **3** → Realiza o *logout* do fisioterapeuta.

Devido à maior complexidade da acção levada a cabo a partir do botão 2 presente na Figura 3.11, esta é mais detalhada na secção 3.3.1.

3.3.1. Pacientes

Clicando no botão Patients, presente na Figura 3.11 com o número 2, o fisioterapeuta é deslacado para a página apresentada na Figura 3.12.

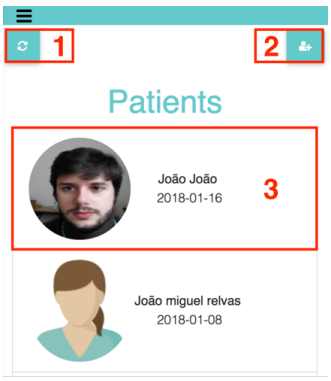


Figura 3.12 – Página dos pacientes na *PhysioWebapp*.

Nesta página o fisioterapeuta pode realizar as seguintes acções:

- **Sincronizar pacientes:** Clicando no Botão 1 da Figura 3.12, o fisioterapeuta pode actualizar a lista de pacientes.
- **Criar pacientes:** Ao clicar no botão 2 da Figura 3.12, o fisioterapeuta é direcionado para o formulário da Figura 3.13, onde pode criar um paciente ou regressar à lista de pacientes premindo o botão 1 presente nessa página.

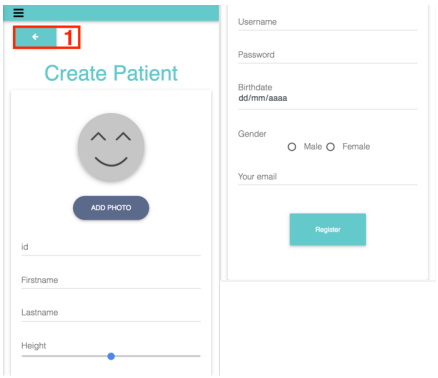


Figura 3.13 – Página de criação do paciente na *PhysioWebapp*.

Para esta criação ser bem-sucedida, o *username* e *email* e *id* escolhidos não devem existir no sistema. Caso não existam será mostrada a mensagem de sucesso da Figura 3.14.

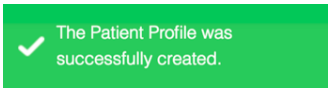


Figura 3.14 – Mensagem de sucesso na criação de um paciente na *PhysioWebapp*.

Por outro lado, caso algum dos dados referidos esteja em uso por outro paciente, será mostrada uma das mensagens de erro da Figura 3.15, sendo que estes devem ser alterados de maneira a finalizar o processo.

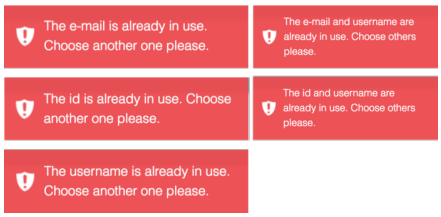


Figura 3.15 – Mensagem de erro na criação de um paciente na *PhysioWebapp*.

- **Selecionar pacientes:** Ao clicar sobre um dos elementos da lista de pacientes da Figura 3.12, o fisioterapeuta é direcionado para perfil do paciente mostrado na Figura 3.16.

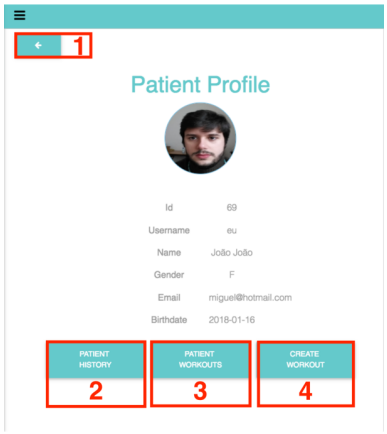


Figura 3.16 – Visualização do perfil de um paciente por parte do fisioterapeuta na *PhysioWebapp*.

No perfil do paciente existem os seguintes botões, numerados de 1 a 4 que permitem ao fisioterapeuta realizar diferentes acções:

1→ Quando premido permite regressar à lista dos pacientes.

2→ Quando premido encaminha o fisioterapeuta para a página com o histórico de sessões do paciente, exibida na Figura 3.17.

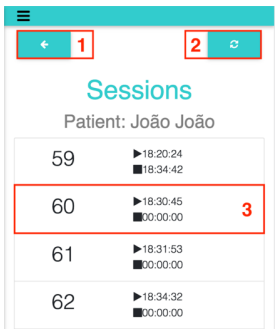


Figura 3.17 – Histórico de sessões de um paciente do fisioterapeuta na PhysioWebapp.

Nesta página o fisioterapeuta pode realizar as seguintes 2 acções:

- **Regresso ao perfil do paciente:** Ao clicar no botão marcado com o número 1 da Figura 3.17.
- **Sincronia das sessões de fisioterapia:** Ao clicar no botão marcado com o número 2 da Figura 3.17.
- **Visualização de uma sessão de fisioterapia:** Clicando numa sessão da lista disponibilizada na Figura 3.17, como a marcada com o número 3.

Figura 3.18 – Página de uma sessão do paciente do fisioterapeuta na *PhysioWebapp*.

Na página de uma sessão de fisioterapia, como a apresentada na Figura 3.18, o fisioterapeuta pode regressar à lista de sessões (botão 1) ou

visualizar os dados relativos à sessão selecionada. Os dados mostrados dividem-se nas seguintes 3 categorias:

- **Dados temporais:** Número da sessão (2); data da sessão (3); hora de início da sessão (4); e hora de término da sessão (5).
- **Métricas calculadas:** Número de passos dados (6); elevação máxima do lado direito (7); elevação máxima do lado esquerdo (8); elevação média do lado direito (9); elevação média do lado esquerdo (10); perímetro de equilíbrio (11); área de equilíbrio (12); ponto de equilíbrio médio do paciente (13); e desvio do equilíbrio do paciente (14);
- **Gráficos da sessão:** Gráfico de orientação (15); gráfico de elevação (16); gráfico de equilíbrio do paciente (17); e o gráfico de forças nos pés do andarilho (18).

3→ Este botão encaminha o fisioterapeuta para a página com o histórico de planos de exercícios do paciente, exibida na Figura 3.19.

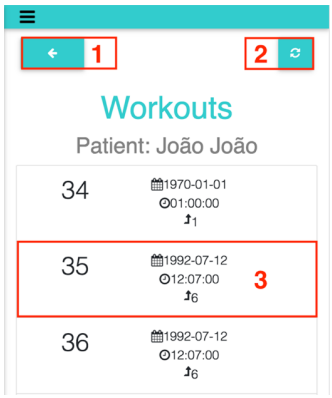


Figura 3.19 – Histórico dos planos de um paciente do fisioterapeuta na *PhysioWebapp*.

Nesta página o fisioterapeuta pode realizar as seguintes acções:

- **Voltar para a página do paciente selecionado:** Clicando no botão 1 da Figura 3.19.
- **Sincronizar os planos de exercício físico:** Clicando no botão 2 da Figura 3.19.

- **Selecionar um plano de exercício físico:** Clicando num plano da lista o fisioterapeuta é direcionado para a página do plano selecionado, exibida na Figura 3.20.

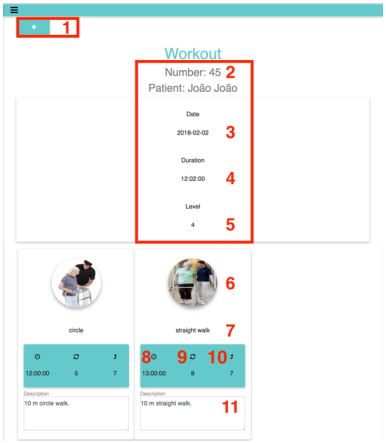


Figura 3.20 – Página de plano de um paciente do fisioterapeuta na *PhysioWebapp*.

Na página de um plano de exercícios, como a apresentada na Figura 3.20, o fisioterapeuta pode regressar à lista de planos (botão 1) ou visualizar os dados relativos ao plano selecionado. Os dados mostrados dividem-se nas seguintes 2 categorias:

- **Dados do plano:** Número do plano (2); data do plano (3); hora de início do plano (4); hora do fim do plano (5).
- **Dados dos exercícios:** Fotografia do exercício (6); título do exercício (7); duração do exercício (8); número de repetições (9); nível de dificuldade do exercício (10); descrição do exercício (11).

4→ Este botão encaminha o fisioterapeuta para a página de criação de um plano de exercícios, exibida na Figura 3.21.

Figura 3.21 – Página de criação de um plano de exercícios na PhysioWebapp.

Nesta página o fisioterapeuta pode regressar aos planos de exercício do paciente, clicando no botão *back* (marcado com o número 1 na Figura 3.21) ou criar um novo plano de exercícios preenchendo e submetendo o formulário presente na mesma figura.

O formulário da criação do plano de exercício divide-se nas seguintes duas categorias:

- **Campos do plano de exercícios:** Estes são o campo da data (2); da duração (3); e da dificuldade do plano de exercício físico a criar (4).
- **Campos de um exercício:** Estes são o campo da fotografia ilustrativa do exercício (7); do título do exercício (8); da duração do exercício (9); do número de repetições do exercício (10); da dificuldade do exercício (11); e da descrição do exercício (12).

Adicionalmente, os botões marcados com o número 5 e 6 na Figura 3.21 podem ser usados para adicionar o retirar exercícios ao plano que está a ser criado.

Finalmente, o botão do marcado com o número 13 na Figura 3.21, é usado para submeter o pedido de criação de um novo plano de exercício. Caso o mesmo seja bem-sucedido é mostrada a mensagem da Figura 3.22.

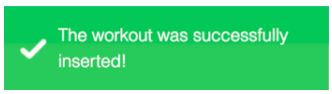


Figura 3.22 – Mensagem de sucesso na criação de um plano de exercício na *PhysioWebapp*.

Caso ocorra algum erro e o plano não tenha sido criado é exibida a mensagem de erro da Figura 3.23.

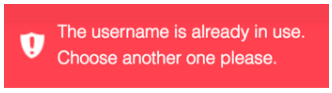


Figura 3.23 – Mensagem de erro na criação de um plano de exercício na *PhysioWebapp*.

3.4. Manual do paciente

Quando um paciente entra na *PhysioWebapp*, este é encaminhado para a sua página de perfil (ver Figura 3.24).

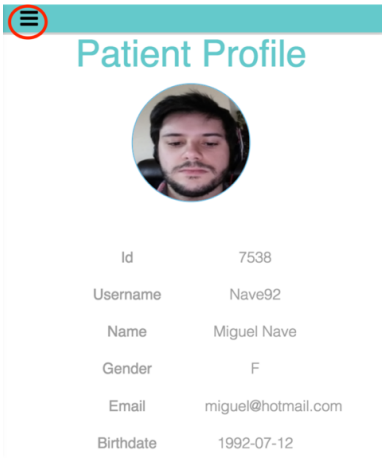


Figura 3.24 – Página de perfil do paciente na *PhysioWebapp*.

No seu perfil, o paciente pode visualizar as suas informações pessoais, assim como clicar no botão marcado a vermelho na Figura 3.24 de modo a aceder ao menu lateral exibido na Figura 3.25.

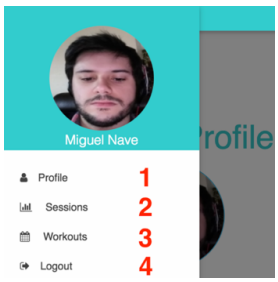


Figura 3.25 – Menu lateral do paciente na *PhysioWebapp*.

Neste menu lateral, o paciente tem à sua disposição os seguintes 4 botões marcados com números de 1 a 4:

- **1** → Encaminha o paciente de novo para o seu perfil.
- **2** → Direciona o paciente para a área das suas sessões.
- **3** → Direciona o paciente para a área dos seus planos de exercício.
- **4** → Realiza o *logout* do paciente.

Devido à maior complexidade da acção levada a cabo a partir dos botões 2 e 3 presentes na Figura 3.25, estas são mais detalhadas nas secções 3.4.1 e 3.4.2, respectivamente.

3.4.1. Área de sessões

Clicando no botão *Sessions*, marcado com o número 2 na Figura 3.25, o paciente é deslocado para a página apresentada na Figura 3.26.

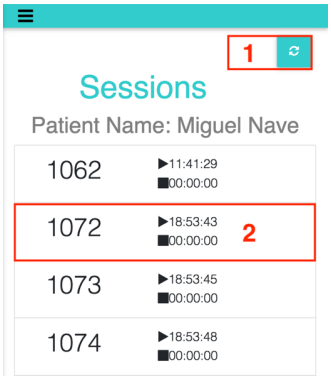


Figura 3.26 – Histórico das sessões de fisioterapia do paciente na *PhysioWebapp*.

Nesta página, o paciente pode realizar as seguintes acções:

- **Sincronizar as sessões de fisioterapia:** Clicando no botão 1 da Figura 3.26, o paciente realiza a sincronização das sessões com o servidor.

- **Selecionar uma sessão de fisioterapia:** Clicando sobre uma sessão da lista, como a marcada com o número 2 na Figura 2.26.

Figura 3.27 – Página de uma sessão de fisioterapia do paciente na *PhysioWebapp*.

Na página de uma sessão de fisioterapia, como a apresentada na Figura 3.27, o paciente pode regressar à lista de sessões de fisioterapia (botão 1) ou visualizar os dados relativos à sessão selecionada. Os dados mostrados dividem-se nas seguintes 3 categorias:

- **Dados temporais:** Número de sessão (2); data da sessão (3); hora de início da sessão (4); hora do fim da sessão (5).
- **Métricas calculadas:** Altura média e máxima do lado esquerdo (6); número de passos (7); altura média e máxima do lado direito (8); área do equilíbrio (9); perímetro do equilíbrio (10); ponto de equilíbrio médio do paciente (11).
- **Gráficos da sessão:** O gráfico de orientação (12); o gráfico de elevação (13); o gráfico de equilíbrio do paciente (14); e o gráfico de forças nos pés do andarilho (15).

3.4.2. Área de planos

Clicando no botão *Workouts*, marcado com o número 3 na Figura 3.25, o paciente é deslocado para a página apresentada na Figura 3.28.

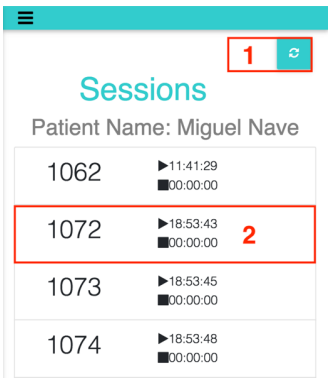


Figura 3.28 – Histórico dos planos de exercício de um paciente na *PhysioWebapp*.

Nesta página, o paciente pode realizar as seguintes acções:

- **Sincronizar os planos de exercício físico:** Clicando no botão 1 da Figura 3.28.
- **Selecionar um plano de exercício físico:** Clicando sobre uma plano da lista como o marcado com o número 2 na Figura 3.28.

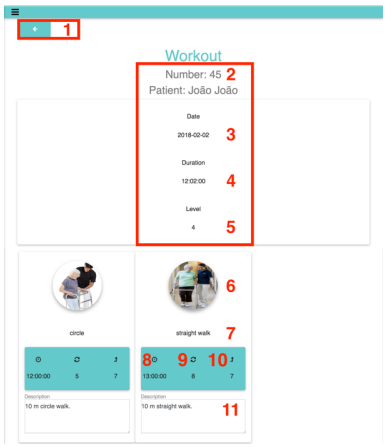


Figura 3.29 – Página de plano de exercício de um paciente na *PhysioWebapp*.

Na página de um plano de exercícios, como a apresentada na Figura 3.29, o paciente pode regressar à lista de planos (botão 1) ou visualizar os dados relativos ao plano selecionado. Os dados mostrados dividem-se nas seguintes 2 categorias:

- **Dados do plano:** Número do plano (2); data do plano (3); duração do plano (4); e nível de dificuldade do plano (5).

- **Dados dos exercícios:** Fotografia do exercício (6); título do exercício (7); duração do exercício (8); número de repetições do exercício (9); nível de dificuldade do exercício (10); descrição do exercício (11).

Capítulo 4 – Manual de instalação

Neste capítulo apresenta-se o processo de instalação das aplicações embebida e móvel do sistema *IoPhyR*. De modo a facilitar este processo é disponibilizada uma *Pen USB* com os dois *softwares*, os ficheiros *PhysioApp.apk* e *andarilho.ino*. Sendo que o primeiro é o ficheiro *APK* da aplicação móvel, e o segundo o código que corre no protótipo de andarilho desenvolvido (ver Figura 4.1).

Figura 4.1 – Conteúdo da *Pen USB* do sistema *IoPhyR*.

4.1. Instalação da aplicação embebida do sistema

A pasta ilustrada na Figura 4.1 exibe o ficheiro *smartwalker.ino* que contém o código presente no protótipo de andarilho desenvolvido. De forma a proceder-se a instalação da aplicação embebida devem ser efectuados os seguintes passos:

- Descarregar o *Arduino IDE* para o computador tendo em conta o sistema operativo do mesmo (ver Figura 4.2).

  Nota: Disponível em https://www.arduino.cc/en/Main/Software.

Figura 4.2 – Opções de transferência para o *Arduino IDE*.

- Abrir o ficheiro *smartwalker.ino*, mostrado na Figura 4.1, com o *Arduino IDE* já descarregado (ver Figura 4.3).

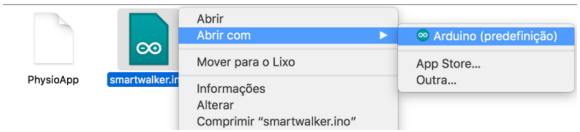


Figura 4.0.3 – Abrir *smartwalker.ino* com *Arduino IDE*.

- Ligar o andarilho ao computador utilizando a entrada *USB* disponível no módulo de computação do mesmo.

- Tendo o ficheiro aberto, ir a Ferramentas. Lá deve ser selecionado em *Placa* "*Arduino/Genuino Mega or Mega 2560*", em Processador "*ATmega2560 (Mega 2560)*" e em *Porta* a porta série que o andarilho está a utilizar (ver Figura 4.4).

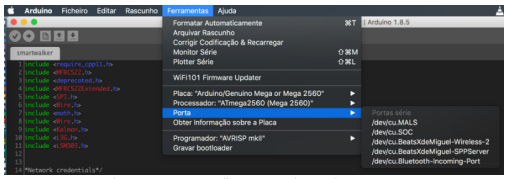


Figura 4.4 – Configuração do *Arduino IDE*.

- Finalmente, deve ser premido o botão selecionado a vermelho na Figura 4.5 de modo a carregar o ficheiro *smartwalker.ino* para o andarilho. Sendo que a barra de progresso, presente no canto inferior direito da janela, deve ficar completa antes de desligar o andarilho do computador.

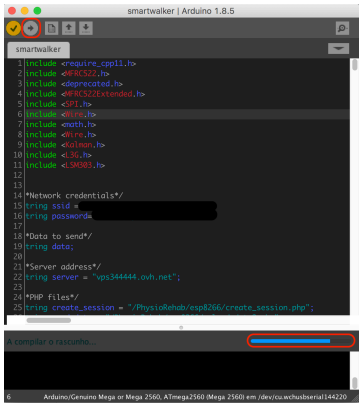


Figura 4.5 – Carregamento da aplicação embebida.

4.2. Instalação da aplicação móvel do sistema

Na pasta apresentada na Figura 4.1 e disponibilizada aos utilizadores do sistema está presente o ficheiro *PhysioApp.apk*. Trata-se de um pacote que deve ser transferido para os dispositivos móveis de modo a proceder à instalação da aplicação *PhysioApp*.

Nota: O dispositivo onde se pretende instalar a *PhysioApp* deve ter a a versão 5.0 do *Android* (*Android Lollipop*) devendo também permitir à ligação à *Internet* e comunicação *Bluetooth*.

A instalação da *PhysioApp* é realizada mediante os seguintes dois passos que devem ser realizados sequencialmente:

Habilitar fontes desconhecidas: Os dispositivos móveis *Android*, por defeito, não permitem instalar aplicações sem ser a partir da *Google Play*. Por isso, e para contornar esse impedimento, deve ser habilitada a intalação de aplicações de fontes desconhecidas, seguindo as seguintes instruções:

- Abrir as *definições* do dispositivo (ver Figura 4.6).

Figura 4.6 – Definições do dispositivo móvel.

- Em *Pessoal*, clicar em *Segurança* (ver Figura 4.7).

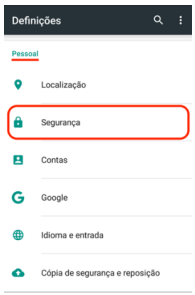


Figura 4.7 – Definições de segurança.

- Em *Segurança* clicar em *fontes desconhecidas* para habilitar a instalação de aplicações fora da *Google Play Store* (ver Figura 4.8).

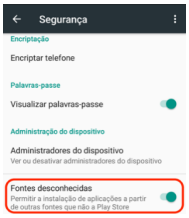


Figura 4.8 – Opção de fontes desconhecidas.

Instalação da *PhysioApp*: Tendo a instalação de aplicações com fontes desconhecidas habilitada, já se pode proceder à instalação da *PhysioApp*. Para isso devem-se realizar os seguintes passos:

- Ligar o dispositivo móvel a um computador utilizando um cabo *USB*, e transferir a *APK*, *PhysioApp.apk* para a pasta *Download* do dispositivo utilizando o explorador de ficheiros (ver Figura 4.9).

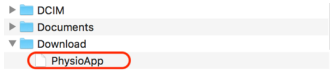


Figura 4.9 – Explorador de ficheiros.

- Abrir o *gestor de ficheiros* do dispositivo móvel (ver Figura 4.10).

Figura 4.10 – Pesquisa por gestor de ficheiros do dispositivo móvel.

- Abrir a pasta de *transferências* do dispositivo móvel (ver Figura 4.11).

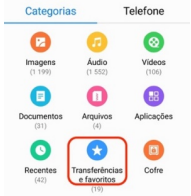


Figura 4.11 – Gestor de ficheiros do dispositivo móvel.

- Clicar na *APK* da aplicação *PhysioApp* dentro da pasta de transferências do dispositivo móvel (Figura 4.12).

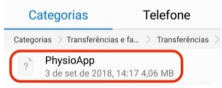


Figura 4.12 – Pasta de transferências do dispositivo móvel.

- Clicar sobre "*aplicações*" para abrir a *APK* como uma aplicação (ver Figura 4.13).

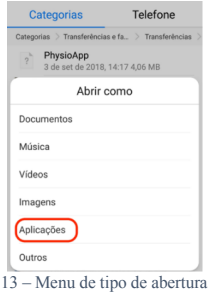


Figura 4.13 – Menu de tipo de abertura da *APK*.

- Clicar no botão *Instalar*, para proceder à intalação da aplicação *PhysioApp* (ver Figura 4.14).

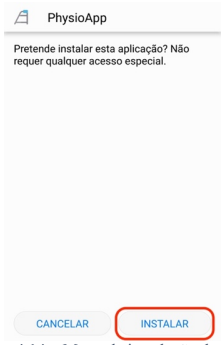


Figura 4.14 – Menu de instalação da *APK*.

- Com a instalação concluída é possível finalizar a mesma premindo no botão com o mesmo nome, ou abrir a aplicação instalada (ver Figura 4.15).

Figura 4.15 – Ecrã de conclusão da instalação da aplicação *PhysioApp*.

Apêndice C – Manual técnico

Instituto Universitário de Lisboa
Departamento de Ciências e Tecnologias da Informação

Manual técnico

IoPhyR – Internet of Physical Rehabilitation IoT System

Carlos Miguel Alpedrinha Ramos de Almeida Nave

Orientador(a):
Doutor Octavian Postolache, Professor Auxiliar,
ISCTE-IUL

Setembro, 2018

IoPhyR – Manual Técnico

i

Índice

Índice de Quadros

IoPhyR – Manual Técnico

v

Índice de Figuras

Descrição do manual técnico

O presente manual técnico tem como objectivo expor e descrever os aspectos técnicos do sistema *IoPhyR* desenvolvido. No primeiro capítulo são abordadas as funções da aplicação embebida do sistema, assim como o seu funcionamento. No segundo capítulo apresenta-se a arquitetura e estrutura do servidor, o *PhysioWatch*, assim como o seu conteúdo e base de dados. No terceiro e último capítulo, são apresentadas as classes e são descritas e explicadas as principais funcionalidades e funções da aplicação móvel desenvolvida, a *PhysioApp*.

Capítulo 1 – Aplicação embebida (Andarilho)

Este capítulo pretende apresentar e descrever as funções principais da aplicação embebida desenvolvida. Esta aplicação foi desenvolvida na linguagem de programação *C*, com o *Arduino IDE 1.8.6*, é usado pelo protótipo de andarilho desenvolvido [1].

1.1. Funções em *C*

A aplicação desenvolida tem as duas seguintes funçõs principais e obrigatórias:

- ***Setup()*:** É executada uma única vez no início do código. Nesta função são realizadas as configurações relativas aos sensores e comunicações (*Wi-Fi*, *Bluetooth* e *Serial*) suportadas pelo sistema.
- ***Loop()*:** É executada continuamente após o término da função *Setup()*. Nesta função é realizado o estabelecimento da comunicação *Bluetooth*, a autenticação do cartão do paciente, o cálculo das métricas necessárias e o envio das mesmas.

A Tabela 1 apresenta o nome, a descrição e categoria de cada uma das funções desenvolvidas para a aplicação embebida.

Nome da função	Descrição	Categoria
protocolsAndPortsConfig()	Realiza a inicialização do protocolo *SPI*, das portas Serial e dos pinos dos sensores de ultrassons.	*setup*
connectWifi()	Tenta estabelecer a comunicação *Wi-Fi* recursivamente.	*setup*
setupElevation()	Realiza o cálculo da distância média dos sensores de ultrassons ao chão quando o andarilho está pousado.	*setup*
avgPressure()	Realiza o cálculo do *offset* de cada um dos 4 sensores de força colocados nos pés do andarilho.	*setup*
initCompass()	Inicializa o sensor *IMU* utilizado e realiza a calibração dos valores por ele obtidos	*setup*
setupLeds()	Inicializa os leds de início e fim da sessão. Liga ainda o led de fim de sessão.	*setup*
checkBluetoothConnection()	Verifica o estabelecimento da comunicação *Bluetooth* enquanto a sessão de fisioterapia não estiver em curso.	*loop*
readPatientCard()	Realiza a leitura de cartões para começar e terminar as sessões de fisioterapia.	*loop*

1

createWifiSession()	Cria uma sessão de fisioterapia a partir da leitura do cartão *RFID* do paciente, utilizando o módulo *Wi-Fi* para realizar pedidos *HTTP*.	*loop*
closeWifiSession()	Termina uma sessão de fisioterapia a partir da leitura do cartão *RFID* do paciente, utilizando o módulo *Wi-Fi* para realizar pedidos *HTTP*.	*loop*
sendHttpPost(*String uri****)***	Realiza pedidos *HTTP* para começar e terminar sessões, assim como para guardar as métricas na base de dados do sistema.	*loop*
resetFunc()	Realiza o *reset* do sistema ao terminar uma sessão	*loop*
showError()	Realiza 10 vezes o piscar do led vermelho quando o cartão *RFID* do paciente não está registado no sistema.	*loop*
metricsCalculations()	Realiza o cálculo das métricas requeridas ao chamar as funções *getEulerAngles()*, *getElevationMetrics()* e *getPressureMetrics()*.	*loop*
getEulerAngles*(String currentTime, String previousTime)*	Calcula os ângulos de orientação, ou de *Euler* (*Roll Pitch* e *Yaw*) usando os dados do acelerómetro, magnetómetro e giroscópio do sensor *IMU*.	*loop*
getElevations()	Calcula a elevação bilateral do andarilho, a partir dos valores de distância dados pelos sensores de ultrassom.	*loop*
getSteps()	Calcula o número de passos dados em função da altura registada pelos sensores de ultrassom.	*loop*
getPressureMetrics()	Calcula a força exercida nos pés do andarilho chamando as funções *getFeetPressure()*, *subtractPressureOffset()*, e o equilíbrio do paciente chamando a função *getBalance()*.	*loop*
getFeetPressure()	Calcula a força lida pelos sensores de força nos pés do andarilho.	*loop*
subtractPressureOffset()	Retira o valor de *offset* previamente calculado aos valores de força obtidos.	*loop*
getBalance()	Calcula o centro de gravidade do paciente a partir da força por este exercida sobre os pé do andarilho.	*loop*
sendWifiMessage()	Envia uma mensagem *HTTP* utilizando a função *sendHttpPost()*.	*loop*
sendBluetoothMessage()	Envia uma mensagem com as métricas calculadas por *Bluetooth*, utilizando uma porta *Serial* e o módulo *Bluetooth*.	*loop*

Tabela 1 – Nomes, descrição e categoria das funções usadas na aplicação embebida.

1.2. Funcionalidades da aplicação

Nesta subsecção apresentam-se e descrevem-se as funcionalidades da aplicação embebida.

- **Comunicação Bluetooth:** Esta comunicação é establecida com os dispositivos móveis com a aplicação do sistema, a *PhysioApp*, de maneira a possibilitar a visualização dos dados da sessão em tempo real. Para isso é usado um módulo *Bluetooth* que funciona através de uma porta *Serial*. Este módulo espera receber um caracter específico (enviado pelo dispositivo móvel) para iniciar a transmissão e outro para a terminar.
- **Comunicação Wi-Fi:** Esta comunicação é estabelecida pelo módulo *Wi-Fi* presente no protótipo de andarilho desenvolvido, de maneira a poder realizar pedidos *HTTP* ao servidor e poder iniciar, terminar ou guardar dados na base de dados do sistema. Esta comunicação é efectuada através de comandos *AT*, quando um cartão é passado no leitor do referido protótipo.
- **Leitura de cartões dos pacientes:** Esta é realizada pelo sensor *RFID* presente no sistema utilizando a biblioteca *mfrc522* [2]. Sendo que a sua utilização serve para autenticar o paciente no início e fim das sessões.
- **Cálculo da orientação:** Este consite no cálculo dos ângulos de orientação, ou ângulos de *Euler* (*Roll*, *Pitch* e *Yaw*), utilizando os dados extraídos do acelerómetro, magnetómetro e giroscópio presentes no sensor *IMU* [3]. Estes cálculos utilizam as bibliotecas *L3G* [4] e *LSM303* [5] disponibilizados pelo fornecedor do sensor. Adicionalmente é usada a biblioteca *KalmanFilter*, que aplica um filtro de *Kalman* de maneira a suavizar os valores dos ângulos obtidos [6].
- **Cálculo da elevação:** Este cálculo, o da elevação bilateral do andarilho, é efectuado com base nos valores adquiridos dos sensores de ultrassons. Estes valores são lidos no início da sessão de maneira a estabelecer um *offset* que é depois retirado no decorrer da mesma.
- **Cálculo do número de passos:** Este cálculo é executado com base nos valores de elevação obtidos, sem *offset*. Sempre e quando é realizada uma transição de elevação de um valor positivo para zero é incrementado o valor de passos dados.
- **Cálculo da força nos pés do andarilho:** Este cálculo é efectuado com base nos valores adquiridos dos sensores de força posicionados nos pés do andarilho. O

valores dados por estes sensores são analógicos (em *Volts*) e para se converterem em *N* são usadas equações características obtidas numa sessão de calibração dos mesmos.

- **Cálculo do equilíbrio:** Este cálculo é efectuado a partir dos valores de força sobre os pés do andarilho, calculados anteriormente. São usadas equações do centro de força que relacionam a posição de cada pé com a força exercida sobre o mesmo.

1.3. Funções principais

Nesta subsecção apresentam-se e explicam-se as funções principais da aplicação embebida.

A função apresentada na Figura 1.1 realiza o cálculo da força exercida sobre cada um dos pés do andarilho. Para isso, primeiramente são guardados em 4 variáveis os valores analógicos adquiridos dos sensores de força colocados nos pés do andarilho (linhas 352 a 355). Estas variáveis adoptam valores no intervalo [0-1023] que tem uma correspondência linear as tensões no intervalo [0-4] *V*. Posteriormente realiza-se a conversão destes valores para *Volts*, utilizando a referida relação entre os valores analógicos e a tensão (linhas 357 a 360). Finalmente são usadas as equações características dos sensores, previamente calculadas, para converter os valores de *Volts* de cada sensor de força em *N (linhas 362-365)*.

Nota: De referir que o valor máximo de tensão de referido é de 4V, e não 5V, devido às características dos sensores e do circuito de condicionamento de sinal usado.

```
350 void getFeetPressure(){
351   /*Get load cells analog raw data*/
352   int sensorValue_FLL = analogRead(A0);
353   int sensorValue_FRL = analogRead(A1);
354   int sensorValue_BRL = analogRead(A2);
355   int sensorValue_BLL = analogRead(A3);
356   /*Convert load cells data to Volts*/
357   float FLL_voltage = sensorValue_FLL * (4.0 / 1023.0);
358   float FRL_voltage = sensorValue_FRL * (4.0 / 1023.0);
359   float BRL_voltage = sensorValue_BRL * (4.0 / 1023.0);
360   float BLL_voltage = sensorValue_BLL * (4.0 / 1023.0);
361   /*Convert load cells data to Newtons*/
362   FLL_pressure = (153.59*FLL_voltage) - 45.061;
363   FRL_pressure = (156.9*FRL_voltage) - 57.786;
364   BRL_pressure = (153.74*BRL_voltage) - 20.648;
365   BLL_pressure = (144.28*BLL_voltage) - 2.5486;
366 }
```

Figura 1.1 – Função do cálculo das forças sobre os pés do andarilho.

A Figura 1.2 apresenta a função *getBalance()* que realiza o cálculo do equilíbrio do paciente, isto é, o centro das forças exercidas nos pés do andarilho. Primeiramente é verificado se estas forças são todas iguais a zero (linha 371). Em caso afirmativo o ponto de equilíbrio do paciente é definido como centrado (linhas 373 e 374). Caso contrário, são calculadas as coordenadas do equilíbrio do paciente, utilizando as equações de centro de forças tendo em conta as forças exercidas sobre os pés do andarilho e as suas coordenadas relativas (linhas 377 a 381).

```
378 void getBalance(){
379     /*forces applied to the walker feet are equal to zero*/
380   if(FLL_pressure==0 && FRL_pressure==0 && BRL_pressure==0 && BLL_pressure==0){
381     /*Balance centered*/
382     centroid_x = 28.1;
383     centroid_y = 20.467;
384   }
385   /*forces applied to the walker feet are different from zero*/
386   else{
387     /*Do the balance calculation*/
388     centroid_x = ((FLL_pressure*6.5)+(FRL_pressure*49.7)+(BRL_pressure*56.2))
389     /(FLL_pressure+FRL_pressure+BRL_pressure+BLL_pressure);
390     centroid_y = ((FLL_pressure*42.8)+(FRL_pressure*42.8))
391     /(FLL_pressure+FRL_pressure+BRL_pressure+BLL_pressure);
392   }
393 }
```

Figura 1.2 – Função do cálculo do equilíbrio do paciente.

A função apresentada na Figura 1.3, a *getElevations()*, realiza o cálculo bilateral do andarilho tendo em conta os valores adquiridos dos sensores de ultrassons usados. Primeiramente é feito o *reset* do pino *trigger* do sensor, utilizando a função *digitalwrite()* para lhe fornecer uma tensão de 0V (linha 226). Posteriormente são aguardados 2 microsegundos utilizando a função *delayMicroseconds()* (linha 228). De seguida são usadas as funções *digitalWrite()* e *delayMicroseconds()* para deixar o pino *trigger* do sensor a *HIGH*, isto é com 5V, durante 10 microsegundos (linhas 230 e 232). Após da passagem deste período, é novamente deixado o pino *trigger* do sensor a *LOW*, isto é, com 0V (linha 234). De seguida, é calculado o tempo de transmissão da onda ultrassónica em microssegundos, utilizando a função *pulseIn()* com valor *HIGH* (5V), no pino *echo* do sensor (linha 236). Finalmente, e tendo em conta o referido tempo de transmissão, a distância do sensor ao chão e a velocidade de transmissão de uma onda ultrassónica, para calcular a elevação do andarilho (linha 238).

Nota: Os cálculos realizados nos intervalos de linhas [226-238] e [241-253] são semelhantes, simplesmente são respectivos a cada um dos sensores.

```
223 float getElevations(){
224   /*---Right side---*/
225     /*Clears the trigPin*/
226     digitalWrite(rightTrigPin, LOW);
227     /*Delay 2 microseconds*/
228     delayMicroseconds(2);
229     /*Sets the trigPin on HIGH state for 10 micro seconds*/
230     digitalWrite(rightTrigPin, HIGH);
231     /*Delay 10 microseconds*/
232     delayMicroseconds(10);
233     /*Clears the trigPin*/
234     digitalWrite(rightTrigPin, LOW);
235     /*Reads the echoPin, returns the sound wave travel time in microseconds*/
236     rightDuration = pulseIn(rightEchoPin, HIGH);
237     /*Calculate the distance*/
238     rightElevation= (rightDuration*0.034/2)-avgElevationRight;
239   /*---Left side---*/
240     /*Clears the trigPin*/
241     digitalWrite(leftTrigPin, LOW);
242     /*Delay 2 microseconds*/
243     delayMicroseconds(2);
244     /*Sets the trigPin on HIGH state for 10 micro seconds*/
245     digitalWrite(leftTrigPin, HIGH);
246     /*Delay 10 microseconds*/
247     delayMicroseconds(10);
248     /*Clears the trigPin*/
249     digitalWrite(leftTrigPin, LOW);
250     /*Reads the echoPin, returns the sound wave travel time in microseconds*/
251     leftDuration = pulseIn(leftEchoPin, HIGH);
252     /*Calculate the distance*/
253     leftElevation= (leftDuration*0.034/2)-avgElevationLeft;
254 }
```

Figura 1.3 – Função do cálculo da elevação bilateral do andarilho.

A Figura 1.4 apresenta a função *getSteps()* que realiza o cálculo do número de passos dados pelo paciente durante uma sessão de fisioterapia, tendo em conta os valores de elevação adquiridos na função *getElevations()*. Primeiramente é verificado se os valores de elevação são superiores a 3 centímetros e se a variável booleana *onStep* é falsa (linha 298). Em caso afirmativo, está a ser dado um passo e por isso a variável *onStep* é verdadeira (linha 300). Caso contrário, isto é, se a variável *onStep* for verdadeira e se as elevações esquerda e direita do andarilho forem iguais a zero, é porque um novo passo foi completado (linha 305). Sendo assim é incrementada a variável que regista o número de passos, a variável *steps*, e é alterado o estado da variável *onStep* para *false* (linhas 307 e 309).

```
296 void getSteps(){
297     /*if there's no steps undergoing and elevation is greater that 3*/
298     if(onStep == false && leftElevation>=3 && rightElevation>=3){
299       /*then a step is undergoing*/
300       onStep = true;
301     }
302     /*if a step is undergoing*/
303     else{
304       /*if a step is undergoing and the elevation is equal to zero*/
305       if(onStep == true && leftElevation==0 && rightElevation==0){
306         /*increment a step to the step variable*/
307         steps++;
308         /*the step is completed*/
309         onStep = false;
310       }
311     }
312 }
```

Figura 1.4 – Função do cálculo dos passos dados pelo paciente.

Finalmente, a função apresentada na Figura 1.5, a *getEulerAngles()*, realiza o cálculo dos ângulos de orientação do paciente, também conhecidos com ângulos de *Euler*, tendo em conta os valores adquiridos do *IMU* utilizado. Primeiramente são realizadas as leituras do giroscópio (linha 403) e do magnetómetro e acelerómetro (linha 405). De seguida é calculado o tempo actual e a orientação do eixo dos *ZZ*, tendo em conta o valor de aceleração no mesmo eixo (linhas 407 a 414). Posteriormente são calculados os valores do acelerómetro, magnetómetro e giroscópio nos três eixos usando os valores extraídos e as características de cada sensor. Sendo que a unidade destes valores é *g* para o acelerómetro, *gauss* para o magnetómetro e *dps* para o giroscópio (linhas 416 a 438). A seguir, são calculados os ângulos de *Euler* (*roll*, *pitch* e *yaw*) utilizando funções trigonométricas e os valores de acelerómetro e magnetómetro (linhas 440 a 444). Finalmente, os ângulos de orientação calculados são suavizados com um filtro de *Kalman* utilizando a biblioteca *KalmanFilter* (linhas 446 a 448).

```
void getEulerAngles(unsigned long currentMillis, unsigned long previousMillis){
    /*Read gyro values*/
    gyro.read();
    /*Read accelerometer and magnetometer values*/
    compass.read();
    /*Get time in seconds*/
    timestamp = currentMillis/1000;
    /*Establish Z axis orientation*/
    if (compass.a.z < 0){
            signOfz = -1;
    }
    else{
            signOfz = 1;
    }

    /* Accelerometer characteristics:
    *16-bit, default range +-2 g, sensitivity 0.061 mg/digit
    */
    /*3 axis acceleration calculus*/
    aX = (double)(compass.a.x)*0.061/1000.0;// g
    aY = (double)(compass.a.y)*0.061/1000.0;// g
    aZ = (double)(compass.a.z)*0.061/1000.0;// g

    /*Magnetometer characteristics:
     * 16-bit, default range +-2 gauss, sensitivity 0.080 mgauss/digit
    */
    /*3 axis magnetometer calculus*/
    mX = (double)(compass.m.x)*0.080/1000.0;// gauss
    mY = (double)(compass.m.y)*0.080/1000.0;// gauss
    mZ = (double)(compass.m.z)*0.080/1000.0;// gauss

    /*Gyroscope characteristics:
    *16-bit, default range +-245 dps (deg/sec), sensitivity 8.75 mdps/digit
    */
    /*3 axis gyroscope calculus*/
    gX = (double)(gyro.g.x)*8.75/1000.0;// dps
    gY = (double)(gyro.g.y)*8.75/1000.0 * signOfz;// dps
    gZ = (double)(gyro.g.z)*8.75/1000.0;// dps

    /*Orientation angles calculus*/
    roll  = asin(-aX, g)*180/M_PI;
    pitch = atan2(aY, aZ)*180/M_PI;
    yaw   = atan2(mY*cos(roll)+mX*sin(pitch)*sin(roll)-mZ*cos(pitch)*sin(roll),
    mX*cos(pitch)+mZ*sin(pitch))*180/M_PI;

    /*Kalman Filter applied to the orientation angles*/
    roll  = kalmanX.getAngle(roll, gX, currentMillis - previousMillis);
    pitch = kalmanY.getAngle(pitch, gY, currentMillis-previousMillis);
}
```

Figura 1.5 – Função do cálculo dos ângulos de orientação.

Capítulo 2 – Servidor *PhysioWatch*

Este capítulo pretende apresentar e descrever a arquitetura, estrutura, organização e o funcionamento do servidor do sistema, o *PhysioWatch*.

2.1. Descrição do servidor

O *PhysioWatch* é o ponto central do sistema *IoPhyR* e tem como incumbência o armazenamento dos dados do sistema para que estes possam ser visualizados sempre que necessário. Entre os dados guardados estão os dados pessoais dos utilizadores, os das sessões de fisioterapia e os dos planos de exercícios.

Em relação à arquitectura, o *PhysioWatch* éum servidor com a configuração *LAMP*, apresentada na Figura 2.1. Sendo o *Linux* o sistema operativo, o *Apache* o servidor web, o *MySQL* o gestor de bases de dados e o *PHP* a linguagem de *scripting* [7].

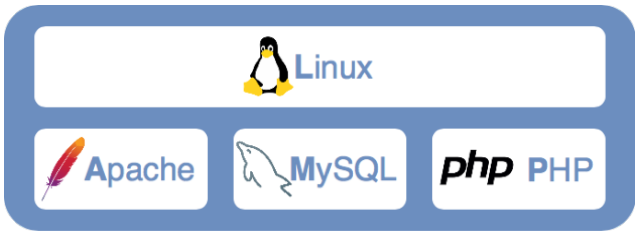


Figura 2.1 – Configuração LAMP.

Existe um conjunto de ficheiros *PHP* criado para que os dados do sistema sejam guardados na base de dados do mesmo, e para que possam ser acedidos pelos utilizadores registados na *PhysioApp*, *PhysioWebapp* e no protótipo de andarilho desenvolvido. Esses ficheiros são acedidos através de pedidos *HTTP POST* transportando os dados requeridos em variáveis (Ver Figura 2.2).

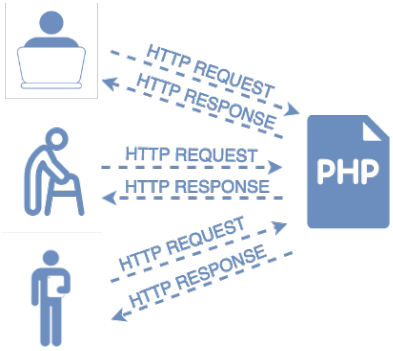


Figura 2.2 – Comunicações HTTP com os ficheiros PHP.

Em relação aos acesso ao servidor, este é feito através de ligações *SSH* (*Secure Shell*) e o acesso e gestão da base de dados é feito através do *phpMyAdmin* em http://vps344444.ovh.net/phpmyadmin.

2.2. Organização e conteúdos do servidor

O *PhysioWatch*, servidor do sistema *IoPhyR*, contém uma base de dados que guarda os dados do sistema, e um conjunto de ficheiros *PHP*. Sendo que a estrutura organizacional das pastas e ficheiros *PHP* do sistema no servidor é a que se apresenta na Figura 2.3. Nesta pasta de raíz encontram-se 3 pastas relativas aos ficheiros *PHP* acedidos pela *PhysioWebapp*, pela *PhysioApp* e pelo andarilho.

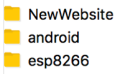


Figura 2.3 – Conteúdos da página raíz do servidor.

- **NewWebsite:** Esta pasta, contém os ficheiros *PHP* das páginas da *PhysioWebapp* que são apresentados na Figura 2.4. Na raíz desta pasta encontram-se a página de login (*login.php*), a de logout (*logout.php*), a de erro (*NotLoggedIn.php*), assim como uma pasta com ficheiros *PHP* que são acedidos por cada um dos tipos de utilizador (administrador, fisioterapeuta e paciente).

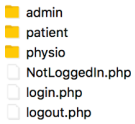


Figura 2.4 – Conteúdo da pasta *NewWebsite* do servidor.

A Figura 2.5, apresenta o conteúdo da pasta *admin*, que aloja os ficheiros *PHP* relativos a secção da *PhysioWebapp* relativa aos admistradores do sistema. Estes ficheiros interagem com a base de dados do sistema e permitem que os administradores realizem as ações que lhes sã permitidas pelo sistema.

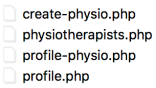


Figura 2.5 – Conteúdo da pasta admin dentro da pasta *NewWebsite* no servidor.

A Figura 2.6, por outro lado, mostra o conteúdo da pasta *patient*, que aloja os ficheiros *PHP* relacionados com secção da *PhysioWebApp* relativa aos

pacientes do sistema. Estas ficheiros permitem aos pacientes visualizarem as informações que lhes são permitidas assim como realizar as funções que lhes são permitidas.

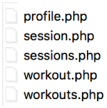


Figura 2.6 – Conteúdo da pasta patient dentro da pasta *NewWebsite* no servidor.

Finalmente, a Figura 2.7, apresenta os ficheiros *PHP* relativos à secção da *PhysioWebapp* para os fisioterapeutas, que estão incluídos na pasta *physio*. Os ficheiros *PHP* incluídos nesta pasta permitem aos fisioterapeutas realizarem as ações que lhes são permitidas pelo sistema.

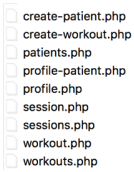


Figura 2.7 – Conteúdo da pasta physio dentro da pasta *NewWebsite* no servidor.

- **android:** Esta pasta contém todos os ficheiros *PHP*, mostrados na Figura 2.8, que permitem que todos os utilizadores do sistema, independentemente do seu tipo, possam interagir com a base de dados a partir da aplicação móvel do sistema. Com estes ficheiros os utilizadores poderão fazer *login*, e realizar as ações que o sistema lhes permite, recebendo ou enviando dados de/para a base de dados do sistema.

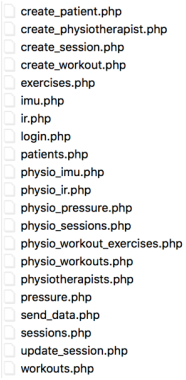


Figura 2.8 – Conteúdo da pasta *android* do servidor.

- **esp8266:** Esta pasta inclui os ficheiros *PHP* que realizam a comunicação entre o andarilho e a base de dados. Os ficheiros registar os dados relativos a sessões realizadas e as que estão em curso (ver Figura 2.9).

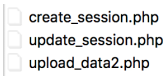


Figura 2.9 – Conteúdo da pasta esp8266 do servidor.

2.3.Base de dados

A base de dados do sistema é uma base de dados *MySQL* devido à sua popularidade e à utilização da arquictura *LAMP*. A sua estrutura foi desenvolvida com a ferramenta *MySQL Workbench* pois permite faze-lo de uma forma visual, sendo posteriormente importada para o *phpMyAdmin*, o gestor de base de dados do sistema [8], [9]. Na Figura 2.10 apresenta-se o diagrama *EER* do base de dados do sistema, que contém informações pessoais dos utilizadores (administradores, fisioterapeutas e pacientes), informações relativas aos planos de exercícios criados e às sessões efectuadas pelos pacientes.

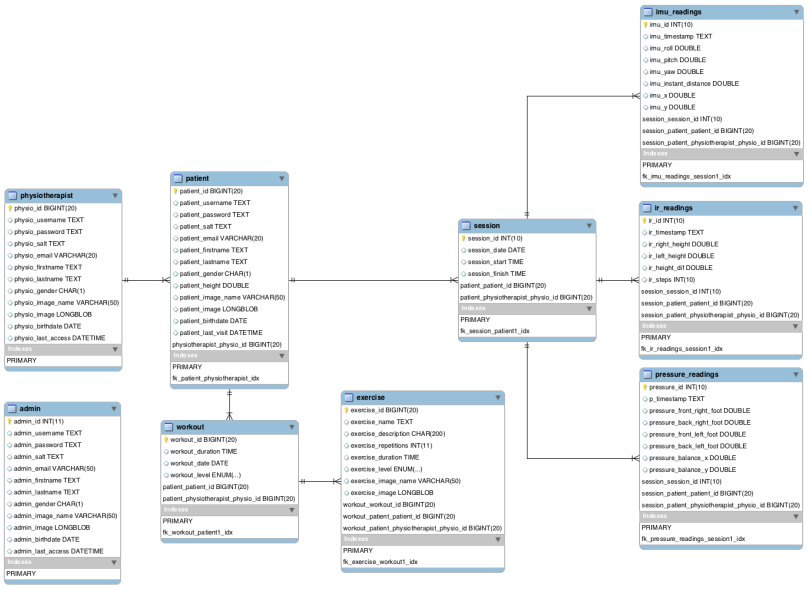


Figura 2.10 – Diagrama EER da base de dados.

O código *SQL* gerado pela estrutura da base de dados criada no *MySQL Workbench*, é apresentado de seguida:

```sql
-- -----------------------------------------------------
-- Schema physiorehabdb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `physiorehabdb` DEFAULT CHARACTER SET latin1 ;
USE `physiorehabdb` ;

-- -----------------------------------------------------
-- Table `physiorehabdb`.`admin`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `physiorehabdb`.`admin` (
  `admin_id` INT(11) NOT NULL AUTO_INCREMENT,
  `admin_username` TEXT NULL DEFAULT NULL,
  `admin_password` TEXT NULL DEFAULT NULL,
  `admin_salt` TEXT NULL DEFAULT NULL,
  `admin_email` VARCHAR(50) NULL DEFAULT NULL,
  `admin_firstname` TEXT NULL DEFAULT NULL,
  `admin_lastname` TEXT NULL DEFAULT NULL,
  `admin_gender` CHAR(1) NULL DEFAULT NULL,
  `admin_image_name` VARCHAR(50) NULL DEFAULT NULL,
  `admin_image` LONGBLOB NULL DEFAULT NULL,
  `admin_birthdate` DATE NULL DEFAULT NULL,
  `admin_last_access` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`admin_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `physiorehabdb`.`physiotherapist`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `physiorehabdb`.`physiotherapist` (
  `physio_id` BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  `physio_username` TEXT NULL DEFAULT NULL,
  `physio_password` TEXT NULL DEFAULT NULL,
  `physio_salt` TEXT NULL DEFAULT NULL,
  `physio_email` VARCHAR(20) NULL DEFAULT NULL,
  `physio_firstname` TEXT NULL DEFAULT NULL,
  `physio_lastname` TEXT NULL DEFAULT NULL,
  `physio_gender` CHAR(1) NULL DEFAULT NULL,
  `physio_image_name` VARCHAR(50) NULL DEFAULT NULL,
  `physio_image` LONGBLOB NULL DEFAULT NULL,
  `physio_birthdate` DATE NULL DEFAULT NULL,
  `physio_last_access` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`physio_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 100000050
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `physiorehabdb`.`patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `physiorehabdb`.`patient` (
  `patient_id` BIGINT(20) UNSIGNED NOT NULL,
  `patient_username` TEXT NULL DEFAULT NULL,
  `patient_password` TEXT NULL DEFAULT NULL,
  `patient_salt` TEXT NULL DEFAULT NULL,
  `patient_email` VARCHAR(20) NULL DEFAULT NULL,
  `patient_firstname` TEXT NULL DEFAULT NULL,
  `patient_lastname` TEXT NULL DEFAULT NULL,
  `patient_gender` CHAR(1) NULL DEFAULT NULL,
  `patient_height` DOUBLE NULL DEFAULT NULL,
  `patient_image_name` VARCHAR(50) NULL DEFAULT NULL,
  `patient_image` LONGBLOB NULL DEFAULT NULL,
  `patient_birthdate` DATE NULL DEFAULT NULL,
  `patient_last_visit` DATETIME NULL DEFAULT NULL,
  `physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
  PRIMARY KEY (`patient_id`, `physiotherapist_physio_id`),
  INDEX `fk_patient_physiotherapist_idx` (`physiotherapist_physio_id` ASC),
  CONSTRAINT `fk_patient_physiotherapist`
    FOREIGN KEY (`physiotherapist_physio_id`)
    REFERENCES `physiorehabdb`.`physiotherapist` (`physio_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `physiorehabdb`.`workout`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `physiorehabdb`.`workout` (
  `workout_id` BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  `workout_duration` TIME NULL DEFAULT NULL,
  `workout_date` DATE NULL DEFAULT NULL,
  `workout_level` ENUM('1', '2', '3', '4', '5', '6', '7', '8', '9', '10') NULL DEFAULT NULL,
  `patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
  `patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
  PRIMARY KEY (`workout_id`, `patient_patient_id`, `patient_physiotherapist_physio_id`),
  INDEX `fk_workout_patient1_idx` (`patient_patient_id` ASC, `patient_physiotherapist_physio_id` ASC),
  CONSTRAINT `fk_workout_patient1`
    FOREIGN KEY (`patient_patient_id` , `patient_physiotherapist_physio_id`)
    REFERENCES `physiorehabdb`.`patient` (`patient_id` , `physiotherapist_physio_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 49
DEFAULT CHARACTER SET = latin1;
```

```
165
166
167   -- -----------------------------------------------------
168   -- Table `physiorehabdb`.`imu_readings`
169   -- -----------------------------------------------------
170   CREATE TABLE IF NOT EXISTS `physiorehabdb`.`imu_readings` (
171     `imu_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
172     `imu_timestamp` TEXT NULL DEFAULT NULL,
173     `imu_roll` DOUBLE NULL DEFAULT NULL,
174     `imu_pitch` DOUBLE NULL DEFAULT NULL,
175     `imu_yaw` DOUBLE NULL DEFAULT NULL,
176     `session_session_id` INT(10) UNSIGNED NOT NULL,
177     `session_patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
178     `session_patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
179     PRIMARY KEY (`imu_id`, `session_session_id`, `session_patient_patient_id`, `session_patient_physiotherapist_physio_id`),
180     INDEX `fk_imu_readings_session1_idx` (`session_session_id` ASC, `session_patient_patient_id` ASC, `session_patient_physiotherapist_physio_id` ASC),
181     CONSTRAINT `fk_imu_readings_session1`
182       FOREIGN KEY (`session_session_id` , `session_patient_patient_id` , `session_patient_physiotherapist_physio_id`)
183       REFERENCES `physiorehabdb`.`session` (`session_id` , `patient_patient_id` , `patient_physiotherapist_physio_id`)
184       ON DELETE NO ACTION
185       ON UPDATE NO ACTION)
186   ENGINE = InnoDB
187   AUTO_INCREMENT = 26563
188   DEFAULT CHARACTER SET = latin1;
189
190
191   -- -----------------------------------------------------
192   -- Table `physiorehabdb`.`ir_readings`
193   -- -----------------------------------------------------
194   CREATE TABLE IF NOT EXISTS `physiorehabdb`.`ir_readings` (
195     `ir_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
196     `ir_timestamp` TEXT NULL DEFAULT NULL,
197     `ir_right_height` DOUBLE UNSIGNED NULL DEFAULT NULL,
198     `ir_left_height` DOUBLE UNSIGNED NULL DEFAULT NULL,
199     `ir_height_dif` DOUBLE UNSIGNED NULL DEFAULT NULL,
200     `ir_steps` INT(10) NULL DEFAULT NULL,
201     `session_session_id` INT(10) UNSIGNED NOT NULL,
202     `session_patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
203     `session_patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
204     PRIMARY KEY (`ir_id`, `session_session_id`, `session_patient_patient_id`, `session_patient_physiotherapist_physio_id`),
205     INDEX `fk_ir_readings_session1_idx` (`session_session_id` ASC, `session_patient_patient_id` ASC, `session_patient_physiotherapist_physio_id` ASC),
206     CONSTRAINT `fk_ir_readings_session1`
207       FOREIGN KEY (`session_session_id` , `session_patient_patient_id` , `session_patient_physiotherapist_physio_id`)
208       REFERENCES `physiorehabdb`.`session` (`session_id` , `patient_patient_id` , `patient_physiotherapist_physio_id`)
209       ON DELETE NO ACTION
210       ON UPDATE NO ACTION)
211   ENGINE = InnoDB
212   AUTO_INCREMENT = 26560
213   DEFAULT CHARACTER SET = latin1;
165
166
167   -- -----------------------------------------------------
168   -- Table `physiorehabdb`.`imu_readings`
169   -- -----------------------------------------------------
170   CREATE TABLE IF NOT EXISTS `physiorehabdb`.`imu_readings` (
171     `imu_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
172     `imu_timestamp` TEXT NULL DEFAULT NULL,
173     `imu_roll` DOUBLE NULL DEFAULT NULL,
174     `imu_pitch` DOUBLE NULL DEFAULT NULL,
175     `imu_yaw` DOUBLE NULL DEFAULT NULL,
176     `session_session_id` INT(10) UNSIGNED NOT NULL,
177     `session_patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
178     `session_patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
179     PRIMARY KEY (`imu_id`, `session_session_id`, `session_patient_patient_id`, `session_patient_physiotherapist_physio_id`),
180     INDEX `fk_imu_readings_session1_idx` (`session_session_id` ASC, `session_patient_patient_id` ASC, `session_patient_physiotherapist_physio_id` ASC),
181     CONSTRAINT `fk_imu_readings_session1`
182       FOREIGN KEY (`session_session_id` , `session_patient_patient_id` , `session_patient_physiotherapist_physio_id`)
183       REFERENCES `physiorehabdb`.`session` (`session_id` , `patient_patient_id` , `patient_physiotherapist_physio_id`)
184       ON DELETE NO ACTION
185       ON UPDATE NO ACTION)
186   ENGINE = InnoDB
187   AUTO_INCREMENT = 26563
188   DEFAULT CHARACTER SET = latin1;
189
190
191   -- -----------------------------------------------------
192   -- Table `physiorehabdb`.`ir_readings`
193   -- -----------------------------------------------------
194   CREATE TABLE IF NOT EXISTS `physiorehabdb`.`ir_readings` (
195     `ir_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
196     `ir_timestamp` TEXT NULL DEFAULT NULL,
197     `ir_right_height` DOUBLE UNSIGNED NULL DEFAULT NULL,
198     `ir_left_height` DOUBLE UNSIGNED NULL DEFAULT NULL,
199     `ir_height_dif` DOUBLE UNSIGNED NULL DEFAULT NULL,
200     `ir_steps` INT(10) NULL DEFAULT NULL,
201     `session_session_id` INT(10) UNSIGNED NOT NULL,
202     `session_patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
203     `session_patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
204     PRIMARY KEY (`ir_id`, `session_session_id`, `session_patient_patient_id`, `session_patient_physiotherapist_physio_id`),
205     INDEX `fk_ir_readings_session1_idx` (`session_session_id` ASC, `session_patient_patient_id` ASC, `session_patient_physiotherapist_physio_id` ASC),
206     CONSTRAINT `fk_ir_readings_session1`
207       FOREIGN KEY (`session_session_id` , `session_patient_patient_id` , `session_patient_physiotherapist_physio_id`)
208       REFERENCES `physiorehabdb`.`session` (`session_id` , `patient_patient_id` , `patient_physiotherapist_physio_id`)
209       ON DELETE NO ACTION
210       ON UPDATE NO ACTION)
211   ENGINE = InnoDB
212   AUTO_INCREMENT = 26560
213   DEFAULT CHARACTER SET = latin1;
```

```
214
215
216 -- -----------------------------------------------------
217 -- Table `physiorehabdb`.`pressure_readings`
218 -- -----------------------------------------------------
219 CREATE TABLE IF NOT EXISTS `physiorehabdb`.`pressure_readings` (
220   `pressure_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
221   `p_timestamp` TEXT NULL DEFAULT NULL,
222   `pressure_front_right_foot` DOUBLE UNSIGNED NULL DEFAULT NULL,
223   `pressure_back_right_foot` DOUBLE UNSIGNED NULL DEFAULT NULL,
224   `pressure_front_left_foot` DOUBLE UNSIGNED NULL DEFAULT NULL,
225   `pressure_back_left_foot` DOUBLE UNSIGNED NULL DEFAULT NULL,
226   `pressure_balance_x` DOUBLE NULL DEFAULT NULL,
227   `pressure_balance_y` DOUBLE NULL DEFAULT NULL,
228   `session_session_id` INT(10) UNSIGNED NOT NULL,
229   `session_patient_patient_id` BIGINT(20) UNSIGNED NOT NULL,
230   `session_patient_physiotherapist_physio_id` BIGINT(20) UNSIGNED NOT NULL,
231   PRIMARY KEY (`pressure_id`, `session_session_id`, `session_patient_patient_id`, `session_patient_physiotherapist_physio_id`),
232   INDEX `fk_pressure_readings_session1_idx` (`session_session_id` ASC, `session_patient_patient_id` ASC, `session_patient_physiotherapist_physio_id` ASC),
233   CONSTRAINT `fk_pressure_readings_session1`
234     FOREIGN KEY (`session_session_id` , `session_patient_patient_id` , `session_patient_physiotherapist_physio_id`)
235     REFERENCES `physiorehabdb`.`session` (`session_id` , `patient_patient_id` , `patient_physiotherapist_physio_id`)
236     ON DELETE NO ACTION
237     ON UPDATE NO ACTION)
238 ENGINE = InnoDB
239 AUTO_INCREMENT = 26560
240 DEFAULT CHARACTER SET = latin1;
```

2.4. Ficheiros *PHP*

O servidor, o *PhysioWatch*, contém um conjunto de ficheiros *PHP* de forma a que se possa realizar interações com a base de dados do sistema *IoPhyR*. Esta interação consiste na realização de pedidos *HTTP POST* que permitem a inserção ou visualização de dados existentes no *PhysioWatch*, como os dados dos utilizadores, das sessões ou dos planos de exercícios.

No que diz respeito à organização dos ficheiros *PHP*, estes estão divididos em função do tipo de "clientes". Isto é, o andarilho, um dispositivo que acede à *PhysioWebapp*, ou à *PhysioApp*. Estes 3 tipos de ficheiros são abordados nas subsecções 2.4.1, 2.4.2 e 2.4.3.

2.4.1. Andarilho

Estes ficheiros *PHP* são usados pelo sistema embebido do sistema, i.e., o andarilho. As funções destes ficheiros são as de criar uma nova sessão de fisioterapia, o envio dos dados das sessões em curso, assim como o término dessas mesmas sessões.

Os ficheiros criados para realizarem as interações do protótipo de andarilho com a base de dados, são os apresentados na Tabela 2.

Nome do ficheiro com o caminho	Descrição
esp8266/create_session.php	Cria uma nova sessão na presente data para o paciente cujo id do cartão foi registado pelo andarilho ao iniciar a prática.
esp8266/upload_data2.php	Insere as métricas calculadas durante a sessão na base de dados do sistema. O ficheiro recebe o id do paciente numa variável para o fazer.
esp8266/update_session.php	Actualiza a hora do término da sessão na base de dados através de um pedido HTTP POST. Para isso utiliza o id do paciente que é transpostado numa variável do pedido.

Tabela 2 – Ficheiros *PHP* do andarilho.

A Figura 2.11 apresenta o ficheiro *esp8266/create_session.php*, acedido pelo andarilho, para que se possa ver em pormenor a interação deste com a base de dados.

```php
<?php
    /*Check used HTTP method and the existance of the patient_id variable*/
    if($_SERVER['REQUEST_METHOD'] == 'POST' AND isset($_POST['patient_id'])){

        /*Database connection*/
        require_once("../connectDB.php");

        /*Store POST method variable*/
        $patient_id = $_POST['patient_id'];

        /*SQL query to store physiotherapist id*/
        $query = "SELECT * FROM patient WHERE patient_id='$patient_id';";
        $resultSet = mysqli_query($con, $query);
        $row = mysqli_fetch_assoc($resultSet);
        $physio_id = $row['physiotherapist_physio_id'];

        /*Set timezone*/
        date_default_timezone_set('Europe/Lisbon');

        /*Get current date and time*/
        $date = date("Y-m-d");
        $time = date("H:i:s");

        /*SQL query to insert new session on the database*/
        $sql = "INSERT INTO session (session_date, session_start, session_finish,
            patient_patient_id, patient_physiotherapist_physio_id)
            VALUES ('$date', '$time', 'NULL','$patient_id', '$physio_id')";

        /*Check if data was sucessfully inserted*/
        if ($con->query($sql) === TRUE) {
            /*Send HTTP 200 OK response*/
            var_dump(http_response_code(200));
        } else {
            /*Send HTTP Unauthorized response*/
            var_dump(http_response_code(401));
        }
    /*The used HTTP method wasn't the required or the HTTP method doesn't have the patient_id variable*/
    } else {
        /*Send HTTP Unauthorized response*/
        var_dump(http_response_code(401));
    }
?>
```

Figura 2.11 – Ficheiro *PHP* para criação de uma sessão de fisioterapia.

O ficheiro apresentado na Figura 2.11, que tem como objectivo a criação de uma nova sessão de fisioterapia, verifica primeiramente se a comunicação *HTTP* realizada com ele mesmo foi establecida com o método *HTTP POST*, e se este transporta com ele a variável *patient_id* (linha 3). Em caso afirmativo o ficheiro prossegue com o seu curso normal, e em caso negativo envia ao dispositivo que iniciou a comunicação uma resposta *HTTP* com o código 401 *Unauthorized*.

Admitindo que o método *HTTP* usado foi o requerido (*POST*), e que a variável *$_POST['patient_id']* existe, segue-se um acesso único ao ficheiro *connectDB.php* que lida com a ligação a base de dados do sistema, e que tem acesso às credenciais de acesso à mesma (linha 6). Posteriormente é guardado na variável local *$patient_id* o identificador do paciente transportado no método *POST* (linha 9). A acção seguinte realiza a extração do *id* do fisioterapeuta, uma vez que será necessário para a criação de uma nova sessão de fisioterapia. Sendo assim, é realizada e executada uma *query SQL* para obter os dados do paciente com identificador igual ao da variável *$patient_id* (linhas 12 e 13), seguida da utilização do método *mysqli_fetch_assoc()*, de forma a estruturar o resultado dessa

query num *array* associativo e guardá-lo na variável local *$row* (linha 14). Finalmente é guardado o *id* do fisioterapeuta em questão na variável local *$physio_id*, acedendo à posição do *array* *$row* relativa ao identificador do fisioterapeuta, a *$row[physiotherapist_physio_id]* (linha 15). Posteriormente é establecido o fuso horário da sessão (linha 18), e são obtidas as horas e a data atuais de maneira a organizar cronologicamente a sessão (linhas 21 e 22). De seguida é realizada a *query* de inserção da nova sessão na base de dados com os dados previamente recolhidos (linhas 25-27), sendo depois executada e verificada a boa execução da mesma. (linhas 30-36). Finalmente, é enviada uma resposta *HTTP* para o dispositivo que efectuou o pedido, que no caso da sessão ter sido criada adopta o código *200 OK* (linha 32), e em caso desta não ter sido autorizada, o código 401 *Unauthorized* (linha 40).

2.4.2. PhysioWebapp

Estes ficheiros *PHP* são usados pela *PhysioWebapp*. Os utilizadores ao entrarem nesta aplicação, podem realizar as acções que lhes são permitidas por intermédio do acesso a ficheiros, que os ligam à base de dados do sistema. Estes ficheiros permitem a criação e visualização de utilizadores e planos de exercício, assim como a visualização dos dados de sessões de fisioterapia.

Os ficheiros PHP criados para o *PhysioWebapp* são os apresentados na Tabela 3.

Nome do ficheiro com o caminho	Descrição
NewWebsite/login.php	Apresenta a página inicial da aplicação *web* com um formulário para que os utilizadores possam realizar o *login* na *PhysioWebapp*.
NewWebsite/logout.php	Realiza o *logout* do utilizador da *PhysioWebapp*.
NewWebsite/NotLoggedIn.php	Avisa o utilizador que tem de realizar o login, quando este tenta aceder a dados sensíveis do sistema e não está logado.
NewWebsite/admin/create-physio.php	Apresenta uma página com um formulário para o administrador poder criar um fisioterapeuta.

NewWebsite/admin/physiotherapists.php	Apresenta uma página com a lista de fisioterapeutas do sistema.
NewWebsite/admin/profile-physio.php	Exibe a página de perfil do fisioterapeuta selecionado na lista de fisioterapeutas pelo administrador na *PhysioWebapp*.
NewWebsite/admin/profile.php	Mostra a página de perfil do administrador quando este realiza um login com sucesso na *PhysioWebapp*.
NewWebsite/physio/create-patient.php	Apresenta a página de criação de um paciente, com um formulário, aos fisioterapeutas.
NewWebsite/physio/create-workout.php	Mostra aos fisioterapeutas a página de criação de um novo plano de exercício físico para os seus pacientes. Sendo que esta criação é feita através de um formulário.
NewWebsite/physio/patients.php	Apresenta uma página com a lista de pacientes do fisioterapeuta que está logado.
NewWebsite/physio/profile-patient.php	Mostra o perfil do paciente selecionado pelo seu fisioterapeuta.
NewWebsite/physio/profile.php	Apresenta uma página com o perfil do fisioterapeuta que entrou na *PhysioWebapp*.
NewWebsite/physio/session.php	Mostra uma página com os dados de uma sessão de fisioterapia realizada por um paciente do fisioterapeuta.
NewWebsite/physio/sessions.php	Apresenta uma página com uma lista com as sessões de fisioterapia realizadas por um paciente do fisioterapeuta que entrou na *PhysioWebapp*.
NewWebsite/physio/workout.php	Mostra uma página com os dados de um plano de fisioterapia atribuído a um

	paciente do fisioterapeuta que entrou na *PhysioWebapp*.
NewWebsite/physio/workouts.php	Apresenta uma página com uma lista com os planos de exercício físico realizados por um paciente do fisioterapeuta que entrou na *PhysioWebapp*.
NewWebsite/patient/profile.php	Apresenta uma página com o perfil do paciente que entrou na *PhysioWebapp*.
NewWebsite/patient/session.php	Mostra uma página com os dados de uma sessão de fisioterapia realizada pelo paciente.
NewWebsite/patient/sessions.php	Apresenta uma página com uma lista com as sessões de fisioterapia realizadas pelo paciente que entrou na *PhysioWebapp*.
NewWebsite/patient/workout.php	Apresenta uma página com uma lista com as sessões de fisioterapia realizadas por um paciente do fisioterapeuta que entrou na *PhysioWebapp*.
NewWebsite/patient/workouts.php	Mostra uma página com os dados de um plano de fisioterapia atribuído ao paciente que entrou na *PhysioWebapp*.

Tabela 3 – Ficheiros *PHP* da *PhysioWebapp*.

2.4.3. PhysioApp

Estes ficheiros servem os utilizadores da aplicação móvel do sistema, a *PhysioApp*. Estes utilizadores, através do acesso a estes ficheiros *PHP*, podem realizar as funções que lhes são permitidas pelo sistema. Estes ficheiros permitem a criação ou visualização de utilizadores, planos de exercício ou sessões, assim como o envio em tempo real de dados de sessões de fisioterapia, e o término destas mesmas sessões.

Os ficheiros criados para realizarem as interações da *PhysioApp* com a base de dados, são os apresentados na Tabela 4.

Nome do ficheiro com o caminho	Descrição
android/create_patient.php	Cria um novo paciente que fica atribuído ao ao fisioterapeuta que requereu a criação do mesmo.
android/create_physiotherapist.php	Realiza a criação de um fisioterapeuta, requerida por um administrador do sistema.
android/create_session.php	Cria um nova sessão em tempo real a pedido de um fisioterapeuta.
android/create_workout.php	Cria um novo plano de exercício físico para o paciente selecionado pelo fisioterapeuta que acede à *PhysioApp*.
android/exercises.php	Envia os dados dos exercícios dos planos dos pacientes para o seus dispositivos móveis (com a *PhysioApp*).
android/imu.php	Envia os dados de orientação de uma sessão de fisioterapia do paciente para o seu dispositivo móvel com a *PhysioApp*.
android/ir.php	Envia os dados de elevação de uma sessão de fisioterapia do paciente para o seu dispositivo móvel (com a *PhysioApp*).
android/login.php	Realiza o login do utilizador na aplicação móvel do sistema, a *PhysioApp*, realizando a verificação das credenciais do mesmo.
android/patients.php	Envia os dados dos pacientes para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physio_imu.php	Envia os dados de orientação de uma sessão de fisioterapia do paciente para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physio_ir.php	Envia os dados de elevação de uma sessão de fisioterapia do paciente para o dispositivo

	móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physio_pressure.php	Envia os dados de força e equilíbrio de uma sessão de fisioterapia do paciente para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physio_sessions.php	Envia os dados das sessões de fisioterapia de um paciente para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physio_workout_exercises.php	Envia os dados dos exercícios dos planos dos pacientes para o dispositivo móvel dos seus fisioterapeutas (com a *PhysioApp*).
android/physio_workouts.php	Envia os dados dos planos de exercícios de um paciente para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*).
android/physiotherapists.php	Envia os dados dos fisioterapeutas para o dispositivo móvel de um administrador (com a *PhysioApp*).
android/pressure.php	Envia os dados de força e equilíbrio de uma sessão de fisioterapia do paciente para o seu dispositivo móvel (com a *PhysioApp*).
android/sessions.php	Envia os dados das sessões de fisioterapia de um paciente para o seu dispositivo móvel (com a *PhysioApp*).
android/workouts.php	Envia os dados dos planos de exercícios de um paciente para o seu dispositivo móvel (com a *PhysioApp*).

Tabela 4 – Ficheiros *PHP* da *PhysioApp*.

A Figura 2.12 apresenta o ficheiro *android/patients.php*, acedido pela *PhysioApp*, para que se possa ver em pormenor a interação deste com a base de dados.

```php
<?php
    if($_SERVER['REQUEST_METHOD']=='POST' AND isset($_POST["username"]) AND isset($_POST["password"])){
        require_once("../connectDB.php");
        $username = $_POST["username"];
        $password = $_POST["password"];

        /*--AUTH--*/
            $query = " SELECT * FROM physiotherapist WHERE physio_username = '$username';";
            $resultSet = mysqli_query($con, $query);

            $row = mysqli_fetch_assoc($resultSet);
            $salt = $row['physio_salt'];
            $physio_id = $row['physio_id'];

            $passSalted = mysqli_real_escape_string($con, $_POST['password']) . $salt;
            $passHashed = hash('sha256', $passSalted);

            $auth = "SELECT * FROM physiotherapist WHERE physio_username = '$username' and physio_password = '$passHashed';";
            $authSet = mysqli_query($con, $auth);
            $num_users = mysqli_num_rows($authSet);
        /*--------*/
        $result = array();

        if($num_users > 0){

            $patientsQuery = "SELECT * FROM patient WHERE physiotherapist_physio_id = '$physio_id';";
            $patientsSet = mysqli_query($con, $patientsQuery);
            $num_patients = mysqli_num_rows($patientsSet);
            /*--------------------*/
            $result[] = array("auth"=>"true","readings"=>$num_patients);
            if($num_patients > 0){
                while($rowSession = mysqli_fetch_array($patientsSet)){
                    $result[] = array(
                            "patient_id"=>$rowSession['patient_id'],
                            "patient_username"=>$rowSession['patient_username'],
                            "patient_email"=>$rowSession['patient_email'],
                            "patient_firstname"=>$rowSession['patient_firstname'],
                            "patient_lastname"=>$rowSession['patient_lastname'],
                            "patient_gender"=>$rowSession['patient_gender'],
                            "patient_height"=>$rowSession['patient_height'],
                            "patient_image_name"=>$rowSession['patient_image_name'],
                            "patient_image"=>$rowSession['patient_image'],
                            "patient_birthdate"=>$rowSession['patient_birthdate'],
                            "patient_last_visit"=>$rowSession['patient_last_visit'],
                            "physiotherapist_physio_id"=>$rowSession['physiotherapist_physio_id']
                            );
                }
            }
        }
        else{
            $result[] = array("auth"=>"false");
        }
    }
    else{
        $result = array();
        $result[] = array("auth"=>"false");
    }
    echo json_encode($result);
?>
```

Figura 2.12 – Ficheiro *PHP* para o envio dos dados dos pacientes de um fisioterapeuta.

O ficheiro apresentado na Figura 2.12, que tem como objectivo o envio dos dados dos pacientes para o dispositivo móvel do seu fisioterapeuta (com a *PhysioApp*), verifica primeiramente se a comunicação *HTTP* realizada com ele mesmo foi establecida com o método *HTTP POST*, e se este transporta com ele as variáveis *username* e *password* (linha 2). Em caso afirmativo o ficheiro prossegue com o seu curso normal, e em caso negativo envia ao dispositivo móvel do fisioterapeuta um array *JSON* com a indicação que a autenticação do fisioterapeuta ou o tipo de pedido não se verificou (linhas 54 a 57).

Admitindo que o método *HTTP* usado foi o requerido (*POST*), e que as variáveis *$_POST['username'] e $_POST['password']* existem, segue-se um acesso único ao ficheiro *connectDB.php* que lida com a ligação a base de dados do sistema, e que tem acesso às credenciais de acesso à mesma (linha 3). Posteriormente são guardados nas variáveis locais *$username* e *$password* respectivamente, o *username* e a *password* do fisioterapeuta que estão presentes nas variáveis *username* e *password*, variáveis essas que

estão encapsuladas no método *POST* (linhas 4 e 5). Posterioremente é pesquisado na base de dados a existência de um fisioterapeuta com as credenciais fornecidas (linhas 8 a 19). De seguida, é obtido o número de pacientes que cumprem essas premissas com a função *mysqli_num_rows()*, que é guardado na variável *$num_users* (linha 20). Depois é criado o *array $result* para a resposta e é verificado se a variável *$num_users* é maior que zero (linhas 22 e 24). Em caso afirmativo, a autenticação do fisioterapeuta é confirmada e o ficheiro prossegue o curso normal. Caso contrário a *array $result* adopta o valor *false* na posição *auth* e é enviado para o fisioterapeuta num ficheiro *JSON* (linhas 50 a 52 e linha 58).

Caso a autenticação seja bem sucedida, é executanda uma *query* na tabela *patient* que procura por pacientes tenham na coluna *physiotherapist_physio_id*, o id do fisioterapeuta que acedeu ao ficheiro *PHP* (linhas 26 a 28). De seguida é preparado o *array result* para o envio, com a posição *auth* a *true*, com o número de pacientes encontrado na posição *readings* (linha 30). Logo após, é verificado se o número de pacientes encontrado é superior a zero. Em caso afirmativo é corrido um ciclo *while* para adicionar todos os pacientes do fisioterapeuta em questão, e os seus dados, ao *array result* (linhas 31 a 48). Finalmente é enviado o referido *array result*, previamente preparado, num ficheiro *JSON* para o fisioterapeuta que realizou o pedido *HTTP* (linha 58).

Capítulo 3 – *PhysioApp*

Este capítulo pretende apresentar o funcionamento da aplicação móvel do sistema, a *PhysioApp*, assim como as classes *Java* que a compõem, as suas características e suas funções principais.

3.1. Descrição

A aplicação móvel do sistema, a *PhysioApp*, é dirigida aos utilizadores do sistema, sejam eles administradores fisioterapeutas e pacientes. A *PhysioApp* requer um *login*, com um *username* e *password*, e permite aos que os seus utilizadores realizem diferentes acções, como se pode ver na Figura 3.1.

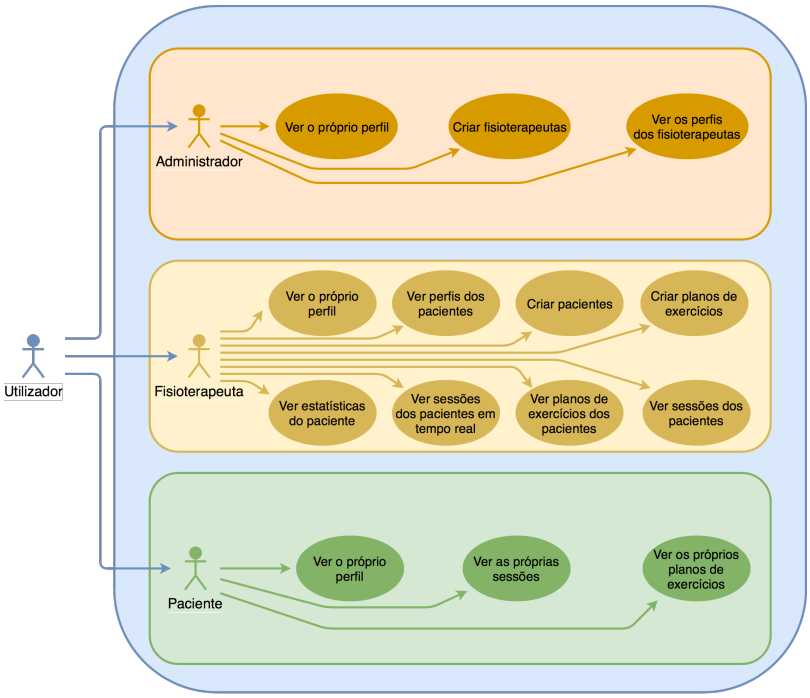


Figura 3.1 – Diagrama de *Use Cases* das funcionalides do sistema.

Esta aplicação foi desenvolvida para o sistema operativo *Android*, tendo a sua versão 5.0 (*Lollipop*) como versão mínima, e requer presença de *Bluetooth* e ligação à *internet* (*Wi-Fi*, *3G*, *LTE* ou *4G*) no dispositivo móvel onde se pretende instalá-la.

3.2. Classes *Java*

A *PhysioApp* é desenvolvida na linguagem de programação *Java* e é constituída por 3 tipos de classes. As do tipo *Activity*, as do tipo *Fragment* e as classes *Java*. As primeiras e as segundas destinguem-se das terceiras pelo facto destas definirem uma "página" da aplicação que é representada por um ficheiro *XML* que lhes está associado [10], [11].
Esta aplicação conta com 4 *Activities*, uma para o login e outras 3, uma para cada tipo de utilizador. Sendo que as dos utilizadores funcionam como "contentores" com um menu lateral, tipo gaveta, que tem o seu conteúdo (as *Fragments*), que é alterado consoante as opções que são selecionadas no referido menu.
A Tabela 5 apresenta o nome, a descrição, e o tipo de todas as classes que constituem a *PhysioApp*.

Nome da classe	Descrição	Tipo
Admin.java	Representa o administrador, guardando os seus dados pessoais. É usado para interagir com a base de dados.	Classe
AdminCreatePhysiotherapeutFragment.java	*Fragment* que permite que o administrador crie um fisioterapeuta.	*Fragment*
AdminPhysiotherapeutProfileFragment.java	*Fragment* que permite ao administrador visualizar os dados de perfil de um fisioterapeuta.	*Fragment*
AdminPhysiotherapeutsFragment.java	*Fragment* que realiza a listagem dos fisioterapeutas ao administrador.	*Fragment*
AdminProfileActivity.java	*Activity* que mostra ao administrador a sua área na aplicação, com um menu lateral para realizar as acções que lhe são permitidas.	*Activity*
AdminProfileFragment.java	*Fragment* que mostra ao administrador o seu perfil com os seus dados pessoais.	*Fragment*
Constants.java	Classe que guarda e encapsula os endereços dos ficheiros *PHP* usados pela *PhysioApp*.	*Classe*
CroppedBitmapImage.java	Classe que permite criar fotografias circulares a partir de um *bitmap*.	Classe
CustomListAdapter.java	*ArrayAdapter* para realizar a listagem dos pacientes numa *ListView* para o fisioterapeuta.	Classe

DataBaseHelper.java	Classe que cria a base de dados da *PhysioApp* e que permite a interação dos utilizadores com a mesma.	Classe
DeviceListAdapter.java	*ArrayAdapter* para realizar a listagem dos dispositivos *Bluetooth* numa *ListView* para o fisioterapeuta.	Classe
Exercise.java	Classe que representa um exercício de um plano de fisioterapia guardando os seus dados. É usada para interagir com a base de dados.	Classe
HistoryFragment.java	*Fragment* disponibiliza ao paciente a listagem das sessões de fisioterapia por si executadas.	*Fragment*
HistorySessionFragment.java	*Fragment* que mostra ao paciente os dados de uma das suas sessões de fisioterapia.	*Fragment*
IMU.java	Classe que representa uma amostra dos dados do sensor *IMU* de uma sessão de fisioterapia específica. É usada para interagir com a base de dados.	Classe
IR.java	Classe que representa uma amostra dos dados do sensores ultrassónicos de uma sessão de fisioterapia específica. É usada para interagir com a base de dados.	Classe
MainActivity.java	*Activity* de entrada na *PhysioApp* que permite aos utilizadores realizarem o login na mesma.	*Activity*
Patient.java	Representa o paciente, guardando os seus dados pessoais. É usado para interagir com a base de dados.	Classe
PatientProfileFragment.java	*Fragment* que mostra ao fisioterapeuta o perfil de um dos seus pacientes.	*Fragment*
PatientsFragment.java	*Fragment* disponibiliza ao fisioterapeuta a listagem dos seus pacientes.	*Fragment*
PhysioCreatePatientFragment.java	*Fragment* que permite que o fisioterapeuta crie um paciente.	*Fragment*

PhysioCreateWorkoutFragment.java	*Fragment* que permite que o fisioterapeuta crie um plano de exercícios para um seu paciente.	*Fragment*
PhysioListAdapter.java	*ArrayAdapter* para realizar a listagem dos fisioterapeutas numa *ListView* para o administrador.	Classe
PhysioPatientProgress.java	*Fragment* que disponibiliza ao fisioterapeuta o progresso de um dos seus pacientes nas últimas 5 sessões de fisioterapia.	*Fragment*
PhysioProfileActivity.java	*Activity* que mostra ao fisiotereapeuta a sua área na aplicação, com um menu lateral para realizar as acções que lhe são permitidas.	*Activity*
PhysioProfileFragment.java	*Fragment* que mostra ao fisioterapeuta o seu perfil com os seus dados pessoais.	*Fragment*
PhysioRealTimeSession.java	*Fragment* que permite ao fisioterapeuta começar uma sessão, e ver os dados da mesma em tempo real.	*Fragment*
PhysioSessionHistoryFragment.java	*Fragment* que mostra ao fisioterapeuta os dados de uma das sessões de fisioterapia realizada por um dos seus pacientes.	*Fragment*
PhysioSessionsHistoryFragment.java	*Fragment* disponibiliza ao fisioterapeuta a listagem das sessões de fisioterapia realizadas por um dos seus pacientes.	*Fragment*
PhysioWorkoutFragment.java	*Fragment* que permite ao fisioterapeuta ver os dados relativos a um dos planos de exercícios atribuídos a um dos seus pacientes.	*Fragment*
PhysioWorkoutsFragment.java	*Fragment* disponibiliza ao fisioterapeuta a listagem dos planos de exercício realizados por um dos seus pacientes.	*Fragment*
Physiotherapist.java	Classe que representa o fisioterapeuta, guardando os seus dados pessoais. É usado para interagir com a base de dados.	Classe

PolygonVertex.java	Classe que representa um ponto com coordenadas *x* e *y* do tipo *Float*.	Classe
Pressure.java	Classe que representa uma amostra dos dados do sensores de força de uma sessão de fisioterapia específica. É usada para interagir com a base de dados.	Classe
ProfileActivity.java	*Activity* que mostra ao paciente a sua área na aplicação, com um menu lateral para realizar as acções que lhe são permitidas.	*Activity*
ProfileFragment.java	*Fragment* que mostra ao paciente o seu perfil com os seus dados pessoais.	*Fragment*
RequestHandler.java	Classe que permite adicionar e processar pedidos *HTTP POST*.	Classe
Session.java	Classe que representa uma sessão de fisioterapia, guardado os dados da mesma. É usada para interagir com a base de dados.	Classe
SessionListAdapter.java	*ArrayAdapter* para realizar a listagem dos das sessões de fisioterapia numa *ListView* para o fisioterapeuta ou paciente.	Classe
SettingsFragment.java	*Fragment* que disponibiliza os dispositivos *Bluetooth* emparelhados e que permite emparelhar o andarilho com o dispositivo móvel ou com o módulo leitor de cartões *RFID*.	*Fragment*
SharedPrefManager.java	Classe que guarda os dados da sessão do utilizador da *PhysioApp*.	Classe
Workout.java	Classe que representa um plano de exercício físico de um paciente, guardando os dados do mesmo. É usada para interagir com a base de dados.	Classe
WorkoutExercisesFragment.java	*Fragment* que permite ao paciente ver os dados relativos a um dos seus planos de exercícios.	*Fragment*
WorkoutListAdapter.java	*ArrayAdapter* para realizar a listagem dos planos de exercícios numa *ListView* para o fisioterapeuta ou paciente.	Classe

WorkoutsFragment.java	*Fragment* que realiza a listagem a um paciente de todos os seus planos de exercício.	*Fragment*

Tabela 5 – Classes *Java* que constituem a *PhysioApp*.

3.3. Características principais

As interfaces desenvolvidas são constituídas pelas seguintes características:

Login: O administrador, fisioterapeuta ou paciente que queira utilizar a *PhysioApp* tem de estar registado no mesmo. O registo dos mesmos é diferente dependendo do tipo de utilizador. O administrador é registado manualmente na base de dados do sistema. O fisioterapeuta é registado pelo administrador, quando este utiliza a aplicação móvel do sistema. Já o registo do paciente é realizado pelo fisioterepeuta ao usar a *PhysioApp*. Para realizar o *login* os utilizadores registados devem inserir as suas credenciais (*username* e *password*) na página de *login*.

Armazenamento local: De modo a não depender do acesso à *internet* para visualizar dados vistos anteriormente, a *PhysioApp* armazena numa base de dados local os dados anteriormente acedidos. É usada uma base de dados *SQLite*, que é *open source,* mais leve que a alojada no servidor, que suporta *SQL*, e usada em multiplataformas, sendo ideal para plataformas móveis. Este tipo de bases de dados contém apenas 5 classes (*NULL*, *INTEGER*, *REAL*, *TEXT*, *BLOB*), que englobam grande parte dos dados existentes numa base de dados *MySQL*. De modo a criar e interagir com esta base de dados é utilizada a classe *SQLiteOpenHelper*, disponibilizada no *Android SDK*.

Comunicação *HTTP*: De modo a interagir com a base de dados do sistema, a *PhysioApp* utiliza pedidos *HTTP* que acedem a ficheiros *PHP* alojados no servidor para o efeito. Para isso é usada a biblioteca *Volley* disponibilizada pela equipa do *Android* [12].

Comunicação *Bluetooth*: Para realizar as sessões de fisioterapia em tempo real e para ler os cartões *RFID* dos pacientes no registo dos mesmos é usada a comunicação *Bluetooth*. Nesse sentido são utilizadas as classes *BluetoothAdapter*, *BluetoothDevice*, *BluetoothSocket* e *BroadcastReceiver* do *Android SDK*. A primeira permite o acesso e uso do módulo *Bluetooth* do dispositivo móvel para começar a descoberta de dispositivos, e listagem de dispositivos emparelhados. A segunda representa o dispositivo com o qual se pretende comunicar e possibilita a ligação *Bluetooth*. A terceira é responsável pela gestão da comunicação bidirecional com os dispositivos (andarilho e módulo leitor de cartões, o *PhysioRegister*) através de um *InputStream* e um *OutputStream*. Finalmente, a quarta permite realizar realizar operações com o módulo *Bluetooth* do dispositivo móvel,

Intents, como ligar e desligar o *Bluetooth*, realizar uma descoberta de novos dispositivos, ver os dispositivos já emparelhados, ou realizar uma comunicação *Bluetooth*.

Desenho dos gráficos: É usada uma biblioteca para elaborar os diversos gráficos que são disponibilizados quer a fisioterapeutas, quer a pacientes. A biblioteca usada é a *MPAndroidChart* [13] que permite desenhar diversos tipos de gráficos, entre eles os usados na aplicação desenvolvida (gráficos de linhas, de área, de barras e de dispersão). Esta biblioteca permite ainda realizar operações de *zoom* dentro dos gráficos.

Imagens circulares: A exibição das fotografias dos utilizadores, e dos exercícios dos planos de exercícios criados é feita utilizando uma biblioteca *Java*. A biblioteca usada é a *CircularImageView* [14] que como o seu nome indica permite criar uma *ImageView* com formato circular, reajustando a imagem sem alterar a qualidade ou as proporções. Deste modo pretende-se tornar a experiência de utilização o mais natural possível, adoptando o formato de imagens mais usual nas aplicações e websites mais populares.

3.4. Funções principais

Nesta secção são apresentadas e descritas algumas das principais funções desenvolvidas na aplicação móvel do sistema, a *PhysioApp*.

3.4.1. Comunicação *HTTP*

As comunicações da aplicação móvel, a *PhysioApp*, com o servidor são feitas através de pedidos *HTTP* que acedem aos ficheiros *PHP* necessários. Sendo que os endereços desses mesmos ficheiros estão guardados na classe *Constants*, mostrada na *Figura 3.2*, para tornar possível essa mesma comunicação.

```java
public class Constants {

    /*<----PHP scripts root path---->*/
    private static final String ROOT_URL = "http://51.255.38.80/PhysioRehab/android/v1/";

    /*<----Login PHP script---->*/
    public static final String URL_LOGIN = ROOT_URL + "login.php";

    /*<----PHP scripts for the patient---->*/
    public static final String URL_GET_SESSIONS = ROOT_URL + "test.php";
    public static final String URL_GET_PRESSURE = ROOT_URL + "pressure.php";
    public static final String URL_GET_IR = ROOT_URL + "ir.php";
    public static final String URL_GET_IMU = ROOT_URL + "imu.php";
    public static final String URL_GET_WORKOUTS = ROOT_URL + "workouts.php";
    public static final String URL_GET_EXERCISES = ROOT_URL + "exercises.php";

    /*<----PHP scripts for the physiotherapist---->*/
    public static final String URL_GET_PATIENTS = ROOT_URL + "patients.php";
    public static final String CREATE_PATIENT = ROOT_URL + "create_patient.php";
    public static final String CREATE_WORKOUT = ROOT_URL + "create_workout.php";
    public static final String PHYSIO_URL_GET_IMU = ROOT_URL + "physio_imu.php";
    public static final String PHYSIO_URL_GET_IR = ROOT_URL + "physio_ir.php";
    public static final String PHYSIO_URL_GET_PRESSURE = ROOT_URL + "physio_pressure.php";
    public static final String URL_PHYSIO_GET_SESSIONS = ROOT_URL + "physio_sessions.php";
    public static final String URL_GET_PHYSIO_WORKOUT_EXERCISES = ROOT_URL + "physio_workout_exercises.php";
    public static final String URL_PHYSIO_GET_WORKOUTS = ROOT_URL + "physio_workouts.php";
    public static final String CREATE_SESSION = ROOT_URL + "create_session.php";
    public static final String SEND_DATA = ROOT_URL + "send_data.php";
    public static final String UPDATE_SESSION = ROOT_URL + "update_session.php";

    /*<----PHP scripts for the administrator---->*/
    public static final String URL_ADMIN_GET_PHYSIOTHERAPISTS = ROOT_URL + "physiotherapists.php";
    public static final String URL_ADMIN_CREATE_PHYSIOTHERAPIST = ROOT_URL + "create_physiotherapist.php";
}
```

Figura 3.2 – Classe Constants com os endereços para os ficheiros PHP.

A função apresentada na Figura 3.3 é uma das que são utilizadas na *PhysioApp*, e é ilustrativa da comunicação *HTTP* da aplicação com o servidor.

```
1   public void getWorkoutsFromServer(){
2       /*HTTP POST Request*/
3       StringRequest stringRequest = new StringRequest(
4               Request.Method.POST,
5               Constants.URL_PHYSIO_GET_WORKOUTS,
6               /*HTTP Response*/
7               new Response.Listener<String>() {
8                   @Override
9                   public void onResponse(String response) {
10                      try{
11                          /*JSON response array*/
12                          JSONArray obj = new JSONArray(response);
13                          String auth = obj.getJSONObject(0).getString("auth");
14                          int number_of_workouts = obj.getJSONObject(0).getInt("readings");
15                          /*check authentication*/
16                          if(auth.equals("true")){
17                              /* If there's no workouts*/
18                              if(number_of_workouts == 0){
19                                  enableOkButton(false);
20                                  showErrorMessage(current_patient.getFirstname()
21                                  + " " + current_patient.getLastname() + " "+ "still don't have any workouts!");
22                              }
23                              /*If there's one or more workouts in the server*/
24                              else{
25                                  /*cicle to iterate the workouts info in the json response*/
26                                  for(int i = 1; i <= number_of_workouts; i++){
27                                      Workout w = new Workout(obj.getJSONObject(i).getInt("workout_id"),
28                                          obj.getJSONObject(i).getString("workout_duration"), obj.getJSONObject(i).getString("workout_date"),
29                                          obj.getJSONObject(i).getInt("workout_level"),obj.getJSONObject(i).getLong("workout_patient_id"),
30                                          obj.getJSONObject(i).getInt("workout_physio_id"));
31                                      /*Add workout to the database*/
32                                      db.addWorkout(w);
33                                  }
34                                  enableOkButton(true);
35                                  /*check if ther's no workouts in the app database*/
36                                  if(!db.workoutEmpty()){
37                                      setWorkoutArrayListAndAdapter(current_patient.getId(), R.layout.workoutslist, workouts_arraylist);
38                                      /*Check if the server has more workouts when compairing with the app database*/
39                                      showWorkoutUpdatedMessage(false);
40                                  }
41                              }
42                          }
43                          else{
44                              /*Show error message if the authentication failed*/
45                              showErrorMessage("Ups! Something went wrong! Your Credentials are wrong!");
46                          }
47                      } catch (JSONException e){
48                          e.printStackTrace();
49                          /*show error message ig the server communication failed*/
50                          showErrorMessage("Ups! There was an error when trying to sync with the server!");
51                      }
52                  }
53              },
54              new Response.ErrorListener(){
55                  /*Show error message if there was an error in the listener*/
56                  @Override
57                  public void onErrorResponse(VolleyError error) {
58                      showErrorMessage("Ups! There was an error when trying to sync with the server!");
59                  }
60              }
61      ){
62          /*Request paramenters*/
63          @Override
64          protected Map<String, String> getParams() throws AuthFailureError {
65              Map<String, String> params = new HashMap<>();
66              params.put("username", username);
67              params.put("password", password);
68              params.put("patient_id", ""+current_patient.getId());
69              return params;
70          }
71      };
72      RequestHandler.getInstance(getContext()).addToRequestQueue(stringRequest);
73  }
```

Figura 3.3 – Comunicação *HTTP* da *PhysioApp* com o servidor.

Esta comunicação é realizada utilizando o objecto *StringRequest* que realiza o pedido *HTTP* definindo o seu método, o endereço do ficheiro *PHP* a aceder, os seus parâmetros e o que fazer ao receber a resposta ao pedido. Os pedidos realizados na aplicação tem o objectivo de receber ou enviar dados de ou para o servidor e só se diferenciam na gestão da resposta ao pedido.

O método do pedido é o *POST* de maneira a encapsular as variáveis enviadas (linha 4). O endereço do ficheiro *PHP* é definido acedendo à classe *Constants* que, como referido anteriormente, contém todos os endereços necessários (linha 5). Os parâmetros do pedido são definidos utilizando a função *getParams()*, onde se cria um *HashMap* com as variáveis a enviar (linhas 61 a 68). Finalmente a resposta ao pedido é gerida no método

onResponse() (linha 9 a 51). Este método recebe como argumento uma *String* que é a resposta ao pedido e que é convertida num objeto *JSONArray* (linha 12). Esse objecto pode conter vários *JSONObject* que verificam a autenticação do utilizador e se contém os dados pedidos (linhas 13 e 14).

3.4.2. Comunicação com a base de dados *SQLite*

A *PhysioApp* conta com uma base de dados *SQLite* de modo a armazenar os dados do sistema no dispositivo móvel, e permitir o acesso *offline* a estes por parte dos utilizadores. A classe *DataBaseHelper* realiza a criação da base de dados e contém métodos que permitem interagir com a mesma para realizar duas operações. Inserir dados novos, e pesquisar dados existentes.

A função apresentada na Figura , a *addSession()*, é ilustrativa da inserção de dados na base de dados. Esta função precisa primeiramente de instanciar um objecto *SQLiteDabase* usando o método *getWritableDatabase()* da classe *DataBaseHelper*, para poder abrir a base de dados em modo de escrita (linha 123) [15]. Depois é necessário criar o *HashMap ContentValues*, para organizar e associar os dados a inserir na tabela às colunas certas (linhas 124 a 130). De seguida é verificado se os dados já existem na tabela, e só caso não existir é que estes são inseridos (linhas 132 a 134). Finalmente é fechada a ligação da base de dados (linha 135).

```
122     public void addSession(Session s){
123         SQLiteDatabase db = this.getWritableDatabase();
124         ContentValues values = new ContentValues();
125         values.put(SESSION_ID, String.valueOf(s.getId()));
126         values.put(SESSION_DATE, String.valueOf(s.getDate()));
127         values.put(SESSION_START, String.valueOf(s.getStart()));
128         values.put(SESSION_FINISH, String.valueOf(s.getFinish()));
129         values.put(SESSION_PATIENT_PATIENT_ID, String.valueOf(s.getPatientId()));
130         values.put(SESSION_PATIENT_PHYSIOTHERAPIST_PHYSIO_ID, String.valueOf(s.getPhysioId()));
131
132         if(!checkIfDataAlreadyInDb(SESSION_TABLE, SESSION_ID, s.getId())) {
133             db.insert(SESSION_TABLE, null, values);
134         }
135         db.close();
136     }
```

Figura 3.4 – Função para adicionar dados à base de dados da *PhysioApp*.

A função apresentada na Figura 3.5, a *getSessionById()*, é ilustrativa da pesquisa de dados na base de dados. Para isso é necessário instanciar um objecto *SQLiteDatabase* utilizando o método *getReadableDatabase()* da classe *DataBaseHelper* [15]. Com este método é possível abrir uma ligação à base de dados em modo de leitura (linha 142). Depois e necessário criar um cursor que recebe o resultado da *query select* necessária (linha 143). Posteriormente é realizado um ciclo *while* para percorrer, estruturar e adicionar o resultado da *query* ao objecto que é devolvido pela função (linha 144 a 154). Finalmente é fechada a ligação da base de dados e é devolvido o objecto requerido (linha 155 e 156).

```
140     public Session getSessionById(int id){
141         Session session;
142         SQLiteDatabase db = this.getReadableDatabase();
143         Cursor cursor = db.rawQuery("select * from session where SESSION_ID = " + id, null);
144         if (cursor.moveToFirst()) {
145             do {
146                 int session_id = cursor.getInt(0);
147                 String date = cursor.getString(1);
148                 String start = cursor.getString(2);
149                 String finish = cursor.getString(3);
150                 long patient_id = cursor.getLong(4);
151                 int physio_id = cursor.getInt(5);
152                 session = new Session(session_id,date,start,finish,patient_id,physio_id);
153             } while (cursor.moveToNext());
154         }
155         cursor.close();
156         return session;
157     }
```

Figura 3.5 – Função para pesquisar dados à base de dados da PhysioApp.

3.4.3. Comunicação *Bluetooth*

A *PhysioApp* realiza comunicações *Bluetooth* com o módulo leitor de cartões, e com o protótipo de andarilho desenvolvido. A primeira é executada para realizar a leitura de cartões dos pacientes aquando do registo destes. Por outro lado a comunicação com o andarilho é feita para proporcionar a visualização dos dados da sessão em tempo real. No entanto o funcionamento para estes dois casos é semelhante. Independentemente do dispositivo com o qual se pretende comunicar, é requerida a criação de uma instância da classe *ConnectedThread*, de dois botões (*start* e *stop*) para a controlar, e de um objecto *Handler* para gerir as mensagens recebidas.

A *ConnectedThread* abre uma *BluetoothSocket* com o dispositivo ao qual se liga. Esta thread conta ainda com um *InputStream* e um *OutputStream* que são usados nos seus métodos *run()* e *write()* para receber e enviar dados para o andarilho ou módulo leitor de cartões. No método *run()* é recebida uma mensagem *Bluetooth* de 2048 *bytes* que é lida e enviada para o *Handler* da *activity* através dos métodos *obtainMessage()* e *sendToTarget()*, identificada com a *tag MESSAGE_READ* (ver Figura 3.6).

```
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;
        // Get the input and output streams, using temp objects because
        // member streams are final
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) { }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }
    public void run() {
        byte[] buffer = new byte[2048];  // buffer store for the stream
        int bytes; // bytes returned from read()
        // Keep listening to the InputStream until an exception occurs
        while (true) {
            try {
                // Read from the InputStream
                if(start == true) {
                    bytes = mmInStream.available();
                    if (bytes != 0) {
                        SystemClock.sleep(100); //pause and wait for rest of data. Adjust this depending on your sending speed.
                        bytes = mmInStream.available(); // how many bytes are ready to be read?
                        bytes = mmInStream.read(buffer, 0, bytes); // record how many bytes we actually read
                        mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer)
                                .sendToTarget(); // Send the obtained bytes to the UI activity
                    }
                }
                else{
                    Log.d("RECEIVER", "Press the start button to start or close session");
                }
            } catch (IOException e) {
                e.printStackTrace();

                break;
            }
        }
    }
    /* Call this from the main activity to send data to the remote device */
    public void write(String input) {
        byte[] bytes = input.getBytes();           //converts entered String into bytes
        try {
            mmOutStream.write(bytes);
        } catch (IOException e) { }
    }
    /* Call this from the main activity to shutdown the connection */
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}
```

Figura 3.6 – *Thread* da comunicação *Bluetooth*.

Os botões criados, *startSession* e *stopSession* controlam a comunicação executando os método *write()* e *cancel()* da *ConnectedThread* criada, quando os botões são clicados. Sendo que para começar a comunicação é enviado o número 1 e para fechar o número 0 (ver Figura 3.7).

```
startSession.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(mConnectedThread != null) { //First check to make sure thread created
                if(start == false) {
                    createSession();
                    mConnectedThread.write("1");
                    start = true;
                }

            }
        }
    });
    stopSession.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(mConnectedThread != null) { //First check to make sure thread created
                if(start == true) {
                    updateSession();
                    mConnectedThread.write("0");
                    start = false;
                    deleteAllEntries();
                    mConnectedThread.cancel();
                }
            }
        }
    });
```

Figura 3.7 – Botões controladores da comunicação *Bluetooth*.

Finalmente o *Handler* instanciado utiliza o método *handleMessage()* que recebe as mensagens do sistema operativo e verifica se alguma delas contém a *tag* que é utilizada pela *ConnectedThread*, a *MESSAGE_READ*. Caso seja recebida uma mensagem com a *tag* referida, esta é lida, dividida e utilizada na *View* ativa da *activity* em questão.

```
mHandler = new Handler(){
    public void handleMessage(android.os.Message msg){
        if(msg.what == MESSAGE_READ){
            String readMessage = null;
            try {
                readMessage = new String((byte[]) msg.obj, "UTF-8");
                String[] message = readMessage.split(";");
                timestamp = Float.valueOf(message[0]);
                float roll = Float.valueOf(message[1]);
                float pitch = Float.valueOf(message[2]);
                float yaw = Float.valueOf(message[3]);
                float distanceRight = Float.valueOf(message[4]);
                float distanceLeft =  Float.valueOf(message[5]);

                float front_right = Float.valueOf(message[6]);
                float back_right = Float.valueOf(message[7]);
                float front_left = Float.valueOf(message[8]);
                float back_left = Float.valueOf(message[9]);

                float center_x = Float.valueOf(message[10]);
                float center_y = Float.valueOf(message[11]);

                addHeightValue(timestamp, distanceRight, distanceLeft);
                addImuEntry(timestamp,roll, pitch, yaw);
                addPressureValue(timestamp, front_left,front_right,back_right,back_left);
                addInstantHeightValue(distanceRight,distanceLeft);
                addGaitValue(center_x,center_y);
                sendDataToServer(timestamp,roll,pitch,yaw,distanceRight,distanceLeft,
                    front_right,front_left,back_right,back_left,center_x,center_y);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
        }
    }
};
```

Figura 3.8 – Objecto *Handler* que gere a mensagem *Bluetooth* recebida.

Bibliografia

[1] Arduino, “Arduino - Software.” [Online]. Available: https://www.arduino.cc/en/Main/Software. [Accessed: 10-Sep-2018].
[2] rfid, “miguelbalboa.” [Online]. Available: https://github.com/miguelbalboa/rfid. [Accessed: 29-Aug-2018].
[3] CH Robotics, “Understanding Euler Angles | CH Robotics.” [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. [Accessed: 29-Aug-2018].
[4] Pololu, “l3g-arduino.” [Online]. Available: https://github.com/pololu/l3g-arduino. [Accessed: 29-Aug-2018].
[5] Pololu, “lsm303-arduino.” [Online]. Available: https://github.com/pololu/lsm303-arduino. [Accessed: 29-Aug-2018].
[6] TKJElectronics, “KalmanFilter.” [Online]. Available: https://github.com/TKJElectronics/KalmanFilter. [Accessed: 29-Aug-2018].
[7] IBM, “Introduction to LAMP technology,” 2005. [Online]. Available: https://www.ibm.com/developerworks/web/tutorials/wa-lamp/wa-lamp.html. [Accessed: 16-Feb-2018].
[8] MySQL, “MySQL :: MySQL Workbench,” 2018. [Online]. Available: https://www.mysql.com/products/workbench/. [Accessed: 16-Feb-2018].
[9] “phpMyAdmin.” [Online]. Available: https://www.phpmyadmin.net/. [Accessed: 28-May-2018].
[10] “Activity | Android Developers.” [Online]. Available: https://developer.android.com/reference/android/app/Activity. [Accessed: 10-Sep-2018].
[11] “Fragments | Android Developers.” [Online]. Available: https://developer.android.com/guide/components/fragments. [Accessed: 10-Sep-2018].
[12] Google, “Volley.” [Online]. Available: https://github.com/google/volley. [Accessed: 29-Aug-2018].
[13] PhilJay, “MPAndroidChart.” [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed: 29-Aug-2018].
[14] Lopspower, “CircularImageView.” [Online]. Available: https://github.com/lopspower/CircularImageView. [Accessed: 29-Aug-2018].
[15] Google, “SQLiteDatabase | Android Developers.” [Online]. Available: https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase. [Accessed: 10-Sep-2018].