ISCTE IUL
Instituto Universitário de Lisboa
Departamento de Ciências e Tecnologias da Informação

Gamificação como Solução para os Problemas da Aprendizagem da Programação

Ricardo Manuel Silva Pereira

Dissertação submetida como requisito parcial para

obtenção do grau de

Mestre em Informática e Gestão

Orientador:

Doutor Carlos Manuel Jorge Costa, Professor Associado

ISEG (School of Economics and Management), Universidade de Lisboa

Outubro, 2017

Resumo

A necessidade de pessoas com conhecimentos de programação é premente. Deste modo a aprendizagem de programação é uma temática particularmente relevantes. Neste documento é abordado o atual estado de arte relativamente ao ensino da programação. São analisados os atuais constrangimentos e desafios que os estudantes têm vindo a enfrentar aquando a aprendizagem da programação, bem como as metodologias adotadas por professores e investigadores de modo a colmatar estas adversidades. Neste contexto, é proposta uma solução que passa pela utilização de gamificação como meio de motivar e incentivar os estudantes a alcançar melhores resultados e a aprender de maneira eficiente. Para o efeito, são descritas as principais características de um sistema gamificado, bem como os princípios da sua implementação. Foi desenvolvido um protótipo demonstrando a exequibilidade da solução proposta. Este protótipo foi ainda sujeito a um período de utilização e validação por parte de uma amostra de utilizadores. Os resultados obtidos sugerem que a utilização de gamificação no contexto da aprendizagem de programação poderá ser benéfico.

Palavras-chave: Gamificação, ensino da programação, aprendizagem.

Abstract

In this thesis we approach the current state of art regarding programming education. We analyze the current constraints and challenges that students have been facing when learning programming, as well as the methodologies adopted by teachers and researchers in order to overcome these adversities. In this context, a solution is proposed that goes through the use of gamification as a means to motivate and encourage students to achieve better results and to learn efficiently. To this end, the main characteristics of a gamified system are described, as well as the principles of its implementation. A prototype was developed demonstrating the feasibility of the proposed solution. This prototype was also subject to a period of use and validation by a sample of users. The results suggest that the use of gamification in the context of programming learning can be beneficial.

Keywords: Gamification, programming teaching, learning.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador professor Carlos Costa por todo o apoio, contribuição e orientação fornecida aquando a realização desta dissertação.

Gostaria também de agradecer à minha namorada, Filipa Pinheiro, e à minha mãe Teresa Ferreira, pelo feedback e apoio que têm fornecido e por estarem sempre disponíveis para analisar e criticar construtivamente o trabalho que tenho desenvolvido.

Aos meus amigos Miguel Baldé e Catarina Martins, por me terem ajudado a desenvolver o protótipo e terem sempre a porta aberta para me apoiar quando mais precisei.

Ao meu grande amigo Mauro Sousa, por ter depositado a sua confiança nas minhas capacidades e me ter incentivo a aceitar este desafio.

Por fim quero agradecer ao meu pai, Manuel Pereira, e aos meus avós, Mamede e Maria Pereira, por me apoiarem e estarem sempre presentes ao longo desta jornada.

Tabela de Conteúdos

Lista de Figuras

Lista de Gráficos

Lista de Tabelas

1. Introdução

1.1 Motivação

O universo tecnológico tem vindo a evoluir exponencialmente, e com essa evolução surgem diversos desafios que devem ser rapidamente identificados e solucionados. Um destes desafios, e provavelmente o mais notório e difícil de colmatar, é a falta de programadores competentes e qualificados para abranger os diversos projetos que vão surgindo no universo das tecnologias de informação. Este facto está intimamente relacionado com a falta de motivação intrínseca que os estudantes têm perante a aprendizagem da programação. Existe, portanto, uma forte necessidade de melhorar as atuais práticas no ensino desta arte, uma vez que a programação não é fácil de aprender, e não se enquadra no perfil da maior parte da população. Para uma correta aprendizagem da programação é necessário trabalhar arduamente e, em regra geral, um processo moroso que envolve a absorção de um conjunto bastante extenso de conceitos, arquiteturas e boas práticas que devem ser ensinadas com o maior rigor para que seja possível a formação de bons programadores (Costa & Aparicio, 2014).

1.2 Delimitação do Tema e Formulação do Problema

Nesta dissertação o principal foco de estudo será a gamificação. Para evitar desvios dos objetivos propostos e para que não hajam julgamentos de valor, iremos focar-nos em dados concretos acerca de estudos realizados em relação à gamificação e a sua aplicabilidade em áreas académicas. O intuito deste trabalho será formular uma resposta concreta em relação ao problema inicialmente identificado:

"Em que medida a gamificação pode contribuir para o aumento da motivação na aprendizagem de programação?"

1.3 Objetivos

Em face do problema proposto, o objetivo principal consiste em:

Identificar em que medida a gamificação pode contribuir para o aumento da motivação na aprendizagem de programação.

Este objetivo pode decompor-se num conjunto de sub-objetivos.

O primeiro objetivo é aumentar o conhecimento acerca da gamificação, uma vez que esta é uma técnica pouco conhecida para a generalidade da população, pelo que, com o desenvolvimento desta dissertação o leitor irá obter um conhecimento mais aprofundado acerca da mesma.

O segundo objetivo é a identificação e análise dos atuais desafios no ensino da programação. É crucial a identificação destes desafios, de modo a alcançar uma solução que os colmate.

Por último, o terceiro objetivo desta dissertação é a proposta de uma solução gamificada como meio de ultrapassar os desafios da aprendizagem de programação identificados no estado de arte.

1.4 Contribuição

Para muitos recém-estudantes de programação, a aprendizagem da mesma é vista como algo bastante difícil e desinteressante que muitas vezes leva ao abandono dos seus cursos, por falta de motivação e também por sentirem que não se encontram na área correta. Esta dissertação poderá contribuir para um melhor desempenho académico dos estudantes de programação, e também pelo aumento do interesse dos mesmos por esta ciência.

O principal contributo desta dissertação será o desenvolvimento de um protótipo gamificado com o intuito de colmatar os principais problemas existentes aquando a aprendizagem de programação, bem como a divulgação dos resultados obtidos após a sua utilização por uma amostra.

1.5 Metodologia

A abordagem metodológica utilizada é *Design Science Research Methodology* (Marsh & Smith, 1995; Jarvinen, 2000; Hevner & Chatteerjee, 2010). Esta metodologia tem como resultados de investigação artefactos. Como fases de investigação esta metodologia apresenta as seguintes fases:

1. Identificação do problema
2. Definição dos objetivos

3. Desenvolvimento e construção de um artefacto
4. Protótipo e uso do artefacto
5. Avaliação do artefacto
6. Comunicação

Nesta dissertação é aqui proposta uma solução conceptual através da análise do atual estado de arte. Para validar a adequabilidade da solução proposta será desenvolvido um protótipo que permitirá analisar em que medida a gamificação poderá aumentar o nível de utilização de um sistema de aprendizagem de programação.

A amostra desta dissertação será um conjunto aleatório de pessoas, definida através da utilização do protótipo, que será divulgado através da rede social Facebook.

A metodologia aplicada na realização desta dissertação será dividida em três fases distintas, cada uma com o seu objetivo específico.

A primeira fase corresponde à leitura de artigos científicos acerca do tema abordado, e realização do estado de arte de modo a atingir dois dos principais objetivos da dissertação, alargar o conhecimento acerca da Gamificação e identificar os principais desafios no ensino da programação.

A segunda fase corresponde ao desenvolvimento de um protótipo que será um *website* com conteúdos de programação, onde os utilizadores poderão aprender e testar os conhecimentos obtidos aquando a utilização do mesmo. Este protótipo terá como principal propósito a obtenção dos dados necessários para atingir o terceiro objetivo, e como tal, irá ter uma vertente gamificada e outra vertente não gamificada, que será estipulada de maneira aleatória de modo a que os dados obtidos não sejam tendenciosos.

A terceira e última fase desta dissertação corresponde à análise dos dados provenientes da utilização do protótipo, que estão armazenados na base de dados, de modo a chegarmos a uma conclusão final acerca do problema inicialmente proposto.

1.6 Estrutura da Dissertação

Esta dissertação está dividida em seis capítulos distintos, cada um com o seu foco específico para que os objetivos inicialmente definidos sejam alcançados.

No primeiro capítulo é formulado o problema inicial, são estabelecidos os objetivos e expostas as motivações e contribuições inerentes à realização deste documento e, por fim, é descrita a metodologia aplicada.

De seguida, no segundo capítulo, é realizado o estado de arte através da análise de trabalhos relacionados de modo a aumentar o conhecimento acerca da gamificação bem como feito o reconhecimento acerca dos principais desafios adjacentes à aprendizagem de programação.

No terceiro capítulo será idealizado um modelo conceptual gamificado como meio de usufruir de todas as capacidades da gamificação e ajudar a combater os desafios identificados no capítulo anterior.

No quarto capítulo iremos detalhar a implementação do protótipo gamificado que propusemos como solução para os problemas da aprendizagem da programação.

No quinto capítulo serão analisados os resultados provenientes da utilização do protótipo e, por fim, no sexto e último capítulo iremos expor as conclusões obtidas com este estudo, e faremos uma análise acerca dos possíveis trabalhos futuros a desenvolver acerca desta temática.

No final deste documento serão indicadas as referências bibliográficas bem como os anexos relativos a esta dissertação.

2. Estado de Arte

2.1 Dificuldades da aprendizagem da programação

De acordo com Combéfis e colegas (2016), é durante o ensino primário e secundário que os estudantes desenvolvem o pensamento algorítmico e computacional necessário para a aprendizagem da programação. Por este motivo os estudantes confrontam-se à partida com um dos grandes desafios da aprendizagem da programação, a compreensão dos seus conceitos básicos, especialmente durante os seus primeiros anos de estudo. Na Holanda, por exemplo, menos de quarenta por cento dos estudantes inscritos em universidades terminaram cursos de três anos em menos de quatro anos (Iosup & Epema, 2014). Segundo Khaleel e colegas (2015), estudos conduzidos a planos curriculares de ciências computacionais e a licenciaturas de engenharia de software concluíram que muitos dos estudantes desistiram no primeiro ou no segundo semestre. Dois dos principais motivos para as desistências de licenciaturas que envolvem unidades curriculares de programação são a tremenda dificuldade que proporcionam, bem como o fato de os estudantes não acharem os seus conteúdos interessantes. Olsson e colegas (2015), acrescentaram ainda que estudos efetuados na Universidade Aberta no Reino Unido indicaram que trinta e oito por cento dos estudantes desistem dos cursos antes de entregarem o seu primeiro trabalho.

Segundo Caspersen e Bennedsen (2007), ensinar programação é considerado um grande desafio e também um dos sete grandes desafios da educação computacional. Estes autores verificaram que alguns estudos realizados apontam para o facto de haver uma perceção de que muitos dos estudantes não conseguem escrever código razoável mesmo após um ou dois semestres de ensino de programação (Caspersen & Bennedsen, 2007), e que por vezes é dito que demora cerca de 10 anos para que um programador novato se torne um programador experiente (Ala-Mutka, 2003).

Existem técnicas que devem ser adotadas quando se quer transmitir com sucesso os conhecimentos relativos à programação, tais como o feedback, a ordenação de tarefas pelo nível de dificuldade e complexidade, personalização dos conteúdos, entre outras. O feedback dos sistemas tem sido também estudado em contexto de comunicação organizacional sendo considerado um fator intangível que assume um papel importante reforço de atitudes (Antunes

& Costa, 2002). Estes são fatores essenciais para uma boa educação (Dicheva, Dichev, Agre & Angelova, 2015). Verdú e colegas (2012), acrescentaram ainda que o fornecimento de apoio instantâneo aos estudantes é um ponto crítico para que a aprendizagem de programação seja efetuada com sucesso, e que o feedback está diretamente relacionado com a aquisição de conhecimentos independentemente das metodologias de ensino aplicadas, e também por aumentar a motivação dos estudantes.

Em 2015, num estudo realizado por Khaleel e colegas (2015), foi verificado que os estudantes geralmente acham difícil aprender uma nova linguagem de programação, uma vez que são postos à prova com novos termos característicos da programação e devido à necessidade de terem de visualizar os processos envolvidos. O resultado desta adversidade é a rutura da mentalidade dos estudantes mais fracos que acabam por achar um fardo todas estas tarefas e acabam por memorizar os processos de programação em vez de tentar compreendê-los, o que conduz à obtenção de maus resultados académicos por parte dos estudantes nestas unidades curriculares. Por outro lado, Vihavainen e colegas (2014), verificaram que os falhanços dos estudantes nesta área não estavam associadas diretamente a uma linguagem de programação específica, mas sim à programação em geral.

Uma das razões pelas quais a aprendizagem de programação é tão árdua, prende-se com o facto de a mesma não poder ser vista como apenas uma aptidão, mas sim como um conjunto complexo de atividades cognitivas, nas quais o estudante têm que simultaneamente construir e aplicar um conjunto de aptidões cognitivas de modo a resolver um problema específico (Vihavainen, Airaksinen & Watson, 2014). Um outro motivo está relacionado com as motivações dos estudantes, onde por vezes, os mesmos não são capazes de criar um modelo mental de como os programas se relacionam com os sistemas que lhes são embebidos, nem criar um modelo acerca dos seus fluxos. Ala-Mutka (2003) acrescentou ainda que os estudantes têm tendência a interiorizar contextos específicos em vez dos contextos gerais o que leva muitas vezes à aplicação incorreta do conhecimento obtido, e como tal, normalmente os estudantes não fazem grandes progressos num curso de programação introdutório. No entanto, muitos estudantes têm a crença de que realmente entenderam o que foi ensinado, apesar de os professores apontarem as suas falhas em trabalhos e exames realizados. Este facto conduz à perceção de que os estudantes não sabem reconhecer as suas fraquezas e deficiências perante a programação.

De acordo com Barata e colegas (2013), quando eram utilizadas metodologias de ensino tradicionais e rudimentares na aprendizagem da programação, como quadros negros, palestras orais, livros e exercícios escritos, existiam problemas ao nível da participação nas ferramentas disponibilizadas pelos professores, havia uma baixa percentagem de estudantes que iam assistir às aulas e também uma falta de interesse nos materiais de referência disponibilizados. Khaleel e colegas (2015), afirmam que uma das possíveis explicações para as dificuldades sentidas pelos estudantes na aprendizagem de programação é a falta de concentração nas aulas, uma vez que os estudantes preferem pensar em temas triviais em vez que de se focarem no que está a ser transmitido pelos professores. Em relação às palestras, muitas vezes são dispensados pontos fulcrais do ensino da programação, fazendo com que os estudantes produzam código desapropriado e repleto de más práticas desde o início da sua aprendizagem. Por estes motivos, foi criada a ideia geral entre os estudantes de que as técnicas e metodologias aplicadas no ensino tradicional de programação são ineficazes e aborrecidas, e portanto os cursos deixam de ser interessantes (Khaleel, Ashaari, Meriam, Wook & Ismail, 2015; Dicheva, Dichev, Agre & Angelova, 2015). Para combater esta tendência e opinião prevalecente entre estudantes, os professores têm vindo a tentar introduzir novas metodologias de ensino. Mas apesar dos seus esforços verifica-se que existe um grande problema ao nível da motivação e empenho demonstrado pelos estudantes (Dicheva, Dichev, Agre & Angelova, 2015).

Verifica-se ainda que nem todas as temáticas desenvolvidas apresentam o mesmo nível de dificuldade (Piteira & Costa, 2012). e que os estudantes e professores têm perceções diferentes relativamente às dificuldades sentidas pelo estudantes (Piteira & Costa, 2013).

2.2 Possíveis soluções para ultrapassar as dificuldades do ensino da programação

A arte de programar é composta por conhecimentos de linguagens e ferramentas de programação, aptidões para resolver problemas e estratégias eficazes de design e implementação de programas (Ala-Mutka, 2003). Uma abordagem comum no ensino da programação é começar por ensinar as bases de uma linguagem de programação e depois ensinar estratégias para tornar os processos da programação eficazes. Portanto, este autor considera importante o foco da aprendizagem inicial em conceitos básicos da programação de

modo a que seja possível desenvolver aptidões mais avançadas entre os estudantes (Ala-Mutka, 2003; Costa, Aparicio & Pierce, 2009; Pierce & Aparicio, 2010).

Segundo Olsson e colegas (2015), desde o início do século vinte e um que está a surgir uma nova tendência em alterar o tradicional ensino face-a-face por cursos em ambiente online com diversas modalidades e atividades, que se não forem bem planeadas podem levar os estudantes a um estado de confusão e solidão, sobretudo em cursos ligados à aprendizagem de programação. Esta metodologia online deverá, tal como a metodologia tradicional, ser planeada com um toque humano de modo a combater o risco de baixar a motivação dos estudantes levando-os a desistir dos seus cursos. Um dos problemas identificados aquando o uso de cursos online é a alta taxa de desistências na fase inicial de implementação, uma vez que pode gerar sobrecarga de conhecimento sobre os estudantes, que por sua vez pode resultar em comportamentos de ansiedade e stress, bem como induzir falta de confiança acrescida, culminando em falhas nos processos de aprendizagem bem como desistências dos cursos.

De acordo com Knutas e colegas (2014), o uso de ferramentas de aprendizagem colaborativas suportadas por computadores está a aumentar, tanto no ensino básico como no ensino superior. Este tipo de ferramentas pode ser utilizado no plano curricular de um curso de ciências computacionais, ou então, como ferramenta de aprendizagem independente a ser explorada pelos estudantes. Vihavainen e colegas (2014), acrescentaram ainda que muitos professores têm dificuldades em ajustar as suas expectativas em relação às capacidades dos seus estudantes, o que conduziu a diversos estudos acerca de como criar e aplicar metodologias de aprendizagem que facilitem o ensino da programação. Estas metodologias visam uma mudança do ensino tradicional baseado em palestras e laboratórios, para uma abordagem mais colaborativa como programação em pares, aprendizagem baseada em jogos (Costa & Aparicio, 2006) e também projetos em equipa. Estas atividades colaborativas, demonstraram ter um impacto significativo na autoconfiança dos estudantes levando-os a melhorar os seus desempenhos e a produzir melhores programas (Verdú, Regueras, Verdú, Leal, de Castro & Queirós, 2012).

Segundo Verdú e colegas (2012), a aprendizagem competitiva conduz a um aumento na participação e empenho dos estudantes através da libertação dos seus instintos competitivos, fator bastante comum em cursos universitários que envolvam programação. No entanto, a autora constatou que existem diferentes sentimentos e motivações entre os vencedores e

perdedores, que podem ser minimizados através de diferentes meios de aplicação deste tipo de metodologia, como por exemplo o foco na aprendizagem e divertimento em vez da vitória, ou na definição clara de critérios de avaliação.

Diversos estudos sugerem que o uso da aprendizagem colaborativa combinada com a aprendizagem competitiva podem aumentar exponencialmente os níveis de motivação e melhorar os desempenhos dos estudantes. No estudo efetuado por Verdú e colegas (2012), foi analisado o impacto das atividades colaborativas em relação ao gênero feminino, onde se constatou que podiam ajudar as mulheres a melhorar a sua representação na área das ciências computacionais, uma vez que as mesmas possuem baixo reconhecimento neste ramo. Por outro lado, observaram que em relação à aprendizagem competitiva, alguns estudos indicam que os homens preferem e sentem-se mais motivados nestes ambientes, no entanto esse impacto parecia não ter efeito nas mulheres.

Segundo Ala-Mutka (2003), um dos métodos que tem sido bastante utilizado no ensino da programação é o uso de visualizações, uma vez que se considera haver benefícios na compreensão de conceitos abstratos e complexos da programação. A maior parte das aplicações desta metodologia foca-se em animações algorítmicas em vez de dar especial atenção à visualização da estrutura base dos programas e das suas execuções. Uma das grandes desvantagens desta metodologia é o tempo necessário para criar, instalar, estudar e integrar as visualizações nos cursos.

Muitos investigadores tentaram mitigar os problemas associados ao ensino da programação através de técnicas como programação de Java baseado em *web*, utilização de animações 3D, aprendizagem baseada em aplicações mobile, aprendizagem baseada em jogos, por exemplo. No entanto não se verificou qualquer melhoria notável, uma vez que os estudantes continuam a não conseguir aprender facilmente as linguagens de programação e portanto têm tendência a chumbar (Khaleel, Ashaari, Meriam, Wook & Ismail, 2015). Para além das formas referidas, muitas outras têm sido apresentadas no contexto da literatura, tal como referido por Costa, Aparicio & Cordeiro (2012 a; 2012 b) ou ainda por Piteira, Costa & Haddad. (2012).

2.3 Gamificação como solução para melhorar a aprendizagem da programação

O termo gamificação foi criado em 2002 por Nick Pelling, mas apenas começou a ganhar notoriedade e popularidade em 2010 (Caponetto, Earp & Ott, 2014; Marczewski, 2013). As primeiras utilizações documentadas da gamificação remontam a 2008, mas o reconhecimento e adoção do termo apenas chegou na segunda metade do ano 2010, quando muitos dos intervenientes neste mercado começaram a sua divulgação (Deterding, Khaled, Nacke & Dixon, 2011) . Segundo Deterding e colegas (2011), gamificação é o uso de elementos dos videojogos para melhorar a experiência, motivação e participação dos utilizadores em aplicações e serviços que não são jogos. Por outro prisma, Koivisto e Hamari consideram que gamificação é o fenómeno de criar experiências jogáveis (Caponetto, Earp & Ott, 2014; Koivisto & Hamari, 2014).

Atualmente existem duas principais ideias em torno do termo gamificação. A primeira é o aumento da adoção e aceitação dos videojogos, bem como a influência que os jogos e os seus elementos têm no nosso quotidiano e nas nossas interações sociais (Deterding, Khaled, Nacke & Dixon, 2011) . O criador de jogos Jesse Schell, sintetizou esta ideia como "Quando a cada segundo da tua vida estás atualmente a jogar um jogo de alguma maneira". A segunda ideia é que uma vez que os videojogos são desenvolvidos para entreter o utilizador em vez de lhe trazer alguma utilidade pura e dura, eles devem produzir efeitos de uma experiência desejável, isto é, cativar os utilizadores e motivá-los a manterem-se numa atividade intensamente durante longos períodos de tempo. Com a adição de elementos dos jogos, é possível criar aplicações, serviços e produtos de grande valor, uma vez que as mesmas se tornam mais apelativas, agradáveis e motivadoras (Deterding, Khaled, Nacke & Dixon, 2011).

Para Deterding e colegas (2011), a gamificação pode ser repartida em diversas componentes que conjugadas conseguem definir na íntegra o conceito de Gamificação: Jogo, Elementos, Contexto não jogável e design. Apesar de os jogos terem o propósito de serem jogados, é importante distinguir a palavra jogo da palavra jogar. Podemos jogar um videojogo, jogar às cartas, jogar futebol ou jogar às escondidas, por exemplo, mas todos estes jogos têm o seu estilo próprio. Em relação à palavra jogar, existem essencialmente duas vertentes que podem ser seguidas, Paidia e Ludus (Groh, 2012; Deterding, Dixon, Khaled & Nacke, 2011). Paidia é

considerado como o estilo de jogo livre, onde cabe aos intervenientes criarem as suas próprias regras em tempo real com as estruturas que bem entenderem num autêntico mundo de fantasia. Por outro lado temos o Ludus, que implica um estilo de jogo controlado com regras e objetivos pré-definidos e as suas respetivas limitações. A gamificação segue a vertente do Ludus, uma vez que se baseia maioritariamente nos componentes de jogos como um sistema de pontuação e definição de regras e objetivos claros, não permitindo muito espaço para a vertente da Paidia, como indicam algumas críticas que foram efetuadas tanto pela indústria como a nível académico. Mas apesar desta limitação, Groh (2012) considera que a gamificação pode transmitir comportamentos e mentalidades associadas à Paidia. Outro fato importante de salientar, é que a maior parte das aplicações gamificadas atualmente existentes foram criadas em formato digital (Deterding, Dixon, Khaled & Nacke, 2011). Esta não deve ser considerada uma limitação, pois a gamificação pode ser aplicada a todo o tipo de jogo como por exemplo um clássico jogo de xadrez.

Enquanto os jogos sérios adotam jogos integrais para propósitos não relacionados com o entretenimento, a gamificação apenas recolhe os elementos desses jogos e aplica-os nesse mesmo contexto (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012). No entanto a distinção entre estes dois conceitos pode ser pessoal, social ou até mesmo subjetiva, uma vez que depende das perceções e doutrinas de cada um (Groh, 2012). Muitas vezes, não é claro se um grupo de pessoas está a jogar ou a utilizar uma aplicação como por exemplo o "Foursquare" (Groh, 2012), que é uma aplicação gamificada onde os seus utilizadores podem disponibilizar a sua localização atual, ou encontrar contactos que se encontrem próximos do seu local atual. Por este motivo, devemos (a) analisar os elementos técnicos e sociais inerentes aos jogos, bem como (b) analisar como os elementos técnicos dos jogos podem proporcionar interpretações e princípios jogáveis em vez serem apenas jogáveis (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012). De acordo com Deterding e colegas (2011), os jogos são uma categoria composta, onde os seus típicos elementos como por exemplo os objetivos e regras, por si só não conseguem constituir um jogo, e muitos deles até já se encontram fora do âmbito dos jogos. Para conseguirmos determinar quais os elementos que se enquadram na categoria de elementos dos jogos, Deterding e colegas (2011) propuseram duas abordagens. A primeira abordagem é liberal onde qualquer elemento encontrado num jogo pode ser considerado, e neste cenário os resultados são ilimitados. Na segunda abordagem, mais restritiva, apenas elementos que são

únicos para os jogos podem ser considerados, e portanto os resultados seriam muito restritos ou até mesmo inexistentes.

Tal como acontece com os jogos sérios, a gamificação utiliza jogos para outros propósitos que vão além do entretenimento (Deterding, Dixon, Khaled & Nacke, 2011). O termo gamificação não deve ser limitado a contextos ou cenários específicos, uma vez que não existem evidências de uma vantagem clara dessa limitação, e também porque as interpretações iniciais levaram muitos autores a focarem-se no contexto da aprendizagem, e a gamificação e os jogos sérios já se proliferaram por diversos outros contextos (Deterding, Dixon, Khaled & Nacke, 2011). Segundo Deterding e colegas (2011), diferentes contextos de utilização devem ser agrupados em subcategorias, tais como gamificação de treino, gamificação de saúde, gamificação das notícias, bem como outras áreas de aplicação. Para além destes factos, Groh (2012) indica que não se deve aplicar gamificação a jogos, uma vez que essa aplicação seria apenas uma extensão do próprio jogo.

Tal como foi constatado anteriormente muitos elementos dos jogos foram utilizados noutros contextos. Analisando aspetos mais técnicos do uso destes elementos em contextos não jogáveis, temos por exemplo a utilização de motores gráficos e ambientes tridimensionais, que costumam ser desenvolvidos para criar videojogos, aplicados a simulações (Groh, 2012). Portanto, Deterding e colegas (2011) propuseram que a gamificação apenas se refira ao desenho dos elementos, em vez de ser considerada uma tecnologia ou prática. Analisando estudos anteriores, Deterding e colegas (2011) chegaram à conclusão de que o desenho dos elementos dos jogos era geralmente descrito tendo como base diversos níveis de abstração, os quais agrupou-os em cinco níveis, ordenados de concretos para abstratos: Padrões de desenho da interface, mecânicas e padrões de desenho de jogo, princípios de desenho de jogo e as suas heurísticas, modelos de jogo e métodos de desenho do jogo.

No que diz respeito aos padrões de desenho da interface geralmente é identificado um problema específico e é desenhada uma solução eficaz, como por exemplo o uso de crachás, tabelas de classificação e níveis (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012).

As mecânicas e padrões de desenho de jogo são as partes constituintes de um jogo que se relacionam com a jogabilidade, tais como as restrições temporais, recursos limitados e jogo por turnos (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012).

Os princípios de desenho de jogo e as suas heurísticas fornecem diretrizes de abordagem a um problema de desenho do jogo ou permitem avaliar uma solução já existente, como por exemplo a existência de objetivos concretos e a diversidade de estilos de jogo (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012).

É importante a criação de modelos conceptuais dos componentes dos jogos ou das suas experiências transmitidas, tal como os desafios, a componente de fantasia e o despertar da curiosidade (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012).

Os métodos de desenho do jogo consistem em práticas e processos específicos do desenho do jogo, como por exemplo a realização de testes a um jogo ou um desenho focado no valor do jogo (Deterding, Dixon, Khaled & Nacke, 2011; Groh, 2012).

O termo gamificação ainda é alvo de críticas por parte de algumas indústrias de videojogos e de media digital, motivando alguns designers e intervenientes neste ramo a alterar o termo para o seu agrado de modo a distanciarem-se das conotações negativas transmitidas por essas mesmas críticas (Deterding, Khaled, Nacke & Dixon, 2011). Existem conceitos semelhantes que estão intimamente relacionados com a gamificação como por exemplo, os jogos sérios, interação lúdica e tecnologias baseadas em jogos (Caponetto, Earp & Ott, 2014; Deterding, Khaled, Nacke & Dixon, 2011). Enquanto a gamificação é a aplicação de componentes dos jogos a um processo que não tem relação alguma com jogos, jogos sérios é a adoção de jogos aplicados ao ensino e são especificamente orientados para que o utilizador aprenda jogando (Caponetto, Earp & Ott, 2014). Apesar de serem duas abordagens possíveis e diferentes, ambas podem ser aplicadas em conjunto (Caponetto, Earp & Ott, 2014).

Em relação aos impactos da gamificação quando aplicada ao ensino da programação, Caponetto e colegas (2014), afirmaram que muitos dos artigos que foram escritos acerca da gamificação apoiam-se em obras anteriormente criadas por outros investigadores, com o objetivo de confirmar a capacidade que a gamificação tem em aumentar a motivação e empenho dos estudantes. Identificaram ainda quais os termos que geralmente estão presentes em artigos acerca da gamificação como é o caso do termo "social" e "design", constatando que pode existir um impacto direto da gamificação sobre os processos de aprendizagem a nível social, e também, da importância do design nas atividades de aprendizagem.

De acordo com Caponetto e colegas (2014), a gamificação é usada para ajudar a completar objetivos transversais tais como fomentar a participação e colaboração entre estudantes,

aprendizagem autodidata, realização dos trabalhos de casa, tornar os processos de avaliações mais fáceis e eficazes e por fim, desenvolver a criatividade dos estudantes. Esta técnica está em crescimento exponencial e a ganhar uma notoriedade elevada sobretudo pela sua utilização em processos de treino e na educação, uma vez que existe a convicção de que a gamificação suporta e motiva os estudantes, bem como melhora os processos de aprendizagem e os seus resultados. Apesar de terem sido realizados diversos estudos acerca da gamificação em diversos níveis educacionais, a tendência e principal foco tem sido o ensino universitário.

No estudo efetuado por Lee e Hammer (2011), foi analisado o que se entendia por gamificação aplicada ao ensino. Estes autores consideraram que a escola deveria ser vista como o maior exemplo de gamificação implicitamente aplicada, uma vez que continha certos elementos semelhantes aos dos jogos. Esclareceram a sua opinião, afirmando que as notas dos estudantes nos seus trabalhos e no final dos anos letivos poderiam ser percecionadas como recompensas por terem sido experienciados comportamentos desejáveis, e que a passagem para o ano letivo seguinte poderia ser considerado como uma subida de nível, no contexto de um jogo. No entanto verificaram que estes elementos, apesar de similares a jogos, não conseguiam ter impactos positivos no empenho dos estudantes, explicando que um possível fato para este acontecimento seria o ambiente padrão transmitido pela própria escola que por vezes fomenta o mau desempenho, uma aprendizagem desamparada, a batota e as desistências.

Os projetos gamificados permitem aos estudantes desenvolver novas opiniões acerca das suas tarefas escolares, uma vez que os mesmos passam a ter a possibilidade de experimentar novas regras, emoções e estatutos sociais, geralmente transmitidos pela gamificação. Através desta experiência, os estudantes passam a ter uma maior motivação e participação no ensino, uma vez que alteram a sua auto-percepção enquanto aprendizes (Lee & Hammer, 2011).

Um ponto crítico na projeção e implementação da gamificação deve ser a definição de desafios concretos que vão ao encontro das aptidões dos estudantes, aumentando o grau de dificuldade consoante os mesmos vão ganhando mais competências (Lee & Hammer, 2011). Através destes desafios os estudantes sentem-se mais motivados a aprender, tal como acontece com os jogos. Outro ponto importante é a possibilidade de escolha do caminho a percorrer para alcançar o sucesso. Para o efeito, os estudantes tendem, com a gamificação, a traçar os seus próprios sub-objectivos de modo a completarem com sucesso as tarefas que lhes foram atribuídas, o que por sua vez também aumenta as suas motivações e empenhos.

No estudo realizado por Ibáñez e colegas (2014), foram analisados os efeitos de gamificação quando aplicada a uma plataforma de questões e respostas acerca da linguagem de programação C. Os resultados obtidos foram benéficos demonstrando que os estudantes continuaram a trabalhar e procuraram aprender mais, mesmo depois de terem obtido a pontuação máxima, o que apoia a ideia de que houve um aumento nas suas motivações. Uma das maiores razões para continuarem empenhados em aprender foi o fato de ainda não terem todos os crachás e de quererem completar essa coleção. Os crachás foram os maiores impulsionadores da participação e quando os estudantes não os conseguiam obter, sentiam-se desencorajados a trabalhar e a aprender. Em relação à tabela classificativa e ao blogue existente na plataforma, os estudantes não acharam atrativa a sua existência. No entanto, não foi possível concluir se a carga de trabalho extra dos estudantes se deveu à existência de gamificação.

2.4 Como gamificar um sistema

Não existe uma definição "per se" de uma metodologia generalizada que possa estar associada à correta implementação de uma aplicação gamificada, no entanto, e segundo Hamari e colegas (2014), foram analisados quais os elementos motivacionais mais utilizados em estudos empíricos e que se encontram intimamente relacionados com a gamificação:

Tabela 2.1: Elementos da Gamificação

Elementos da Gamificação	
Pontos	Tabelas de Classificação
Realizações/Crachás	Níveis
História/Tema	Objetivos Claros
Feedback	Recompensas
Progresso	Desafio

Apesar da existência de diversos estudos relativos à gamificação, Hamari e colegas (2014) verificaram que os pontos, tabelas de classificações e crachás eram sem dúvidas os elementos mais utilizados pelos investigadores. Por outro lado, a empresa Gartner Inc. salienta quais são as quatro principais características que são motivadas pelo uso de um sistema gamificado e que

devem estar presentes no mesmo: (a) Rápido feedback, (b) Objetivos e regras de jogo bem definidas, (c) Narrativa cativante e (d) Tarefas desafiantes mas concretizáveis.

Geralmente o feedback é feito em longos períodos de tempo, mas com o uso da gamificação, o utilizador sabe instantaneamente em que situação se encontra, o que por sua vez motiva a sua participação e continuidade. Os utilizadores sabem à partida o que devem alcançar de modo a subir de estatuto, uma vez que as regras e objetivos ficam claros e percetíveis desde o início da utilização do sistema gamificado. Nas atividades rotineiras que encontramos na nossa sociedade por vezes não somos cativados a abraçá-las, mas com a introdução da gamificação e sobretudo de uma narrativa cativante, os utilizadores tendem a participar mais e a querer atingir os objetivos propostos. Normalmente os desafios costumam ser duradouros, mas a gamificação propõe o uso de tarefas de curta duração, desafiantes e concretizáveis de modo a manter a participação dos seus utilizadores (Rodrigues, Costa & Oliveira, 2013; Rodrigues, Oliveira & Costa, 2016 a; 2016 b) .

Em primeiro lugar Groh (2012) indica que tanto Schell, como Deterding, começam por criticar o estado de arte da gamificação, uma vez que são usados guias para aplicar pontos, crachás e tabelas classificativas em tudo o que se faz, o que pode trazer impactos negativos. Como exemplo desse impacto negativo, foi analisado um estudo que demonstrava que as crianças iriam fazer mais desenhos, mas com menos qualidade, caso fossem pagos para o fazer. Verificou-se ainda que as crianças passaram a gostar menos de desenhar, assim que deixaram de ser pagas. Este efeito é conhecido como "Sobre Justificação", onde as motivações intrínsecas perdem valor e passam a prevalecer as motivações extrínsecas. Surgiram então três princípios que descrevem as necessidades inerentes à motivação intrínseca e que foram formalizadas através de uma íntima comparação com a "Teoria da Auto-Determinação" proposta por Deci e Ryan (1985): Relacionamento, Competência e Autonomia.

Para que um sistema de reputação produza qualquer tipo de impacto, é importante conectar os utilizadores a uma comunidade significativa com os mesmos interesses. Um crachá ou um lugar no topo da tabela classificativa apenas têm significado se for possível exibi-lo a uma comunidade com interesses idênticos, senão perde o seu verdadeiro valor e intuito. Um dos principais cuidados que se deve ter é o contexto social, uma vez que para certas comunidades o objetivo de uma plataforma pode ser claro, para pessoas fora daquele contexto específico de atividade podem ser geradas confusões e ambiguidades (Groh, 2012).

O designer de jogos Ralph Koster (2013) constatou que os jogos transmitem diversão através da sua compreensão e resolução, como por exemplo a resolução de puzzles (Groh, 2012; Koster, 2013). De acordo com Groh (2012), atividades como a resolução de equações seriam divertidas e satisfatórias quando realizadas através de um jogo, uma vez que os seus intervenientes não teriam a noção que estavam a praticar matemática. Por sua vez, o mesmo não se verificaria se fosse aplicado no contexto da educação onde estas mesmas atividades seriam percecionadas como repetitivas e aborrecidas. Este autor concluiu portanto que é importante apresentar desafios interessantes aos utilizadores através da definição de regras e objetivos bem definidos, muitas vezes associando aspetos visuais relevantes que promovam essa perceção. Outro fator importante é a repartição de um objetivo claro em diversas sub-tarefas mais pequenas e concretizáveis, que iriam aumentar de dificuldade consoante a experiência e aptidão do utilizador fosse aumentando (Groh, 2012). Desta maneira seria possível ir ao encontro da "Teoria de fluxo" proposta por Csikszentmihalyi (1990), que descreve o estado mental de uma pessoa perante uma atividade (Groh, 2012; Csikszentmihalyi, 1990), promovendo um equilíbrio entre o desafio e o tédio, onde os intervenientes não se sentiriam demasiado subestimados nem demasiado desafiados. Groh (2012) considera também que o falhanço pode ser visto como algo desejável, uma vez que pode ajudar na experiência de evolução perante o desafio da resolução de um problema, tendo como única adversidade a possível obrigação do utilizador realizar a mesma tarefa demasiadas vezes. O feedback também é visto como uma vantagem do uso dos jogos, uma vez que é fornecido quase instantaneamente, ajudando a encorajar o seus utilizadores, algo que não acontece na vida real.

A maior parte dos jogos segue uma filosofia de jogo voluntário onde a escolha de jogar é intrínseca. Devemos ter cuidado acerca da aplicação de recompensas extrínsecas, como incentivos monetários, sobretudo no contexto do trabalho uma vez que podem ser produzidos efeitos adversos como a sensação de manipulação e perda de autonomia, que por sua vez geram desmotivação. Outro impacto negativo que por vezes ocorre quando usamos motivações extrínsecas é a desvalorização de uma determinada atividade (Groh, 2012).

2.5 Gamificação aplicada a Sistemas de Gestão de Aprendizagem

No estudo realizado por Dicheva e colegas (2015), foi concluído que existem muitas publicações acerca da gamificação, mas que o principal foco tem sido na análise e descrição de mecanismos e dinâmicas de jogo em contexto educacional, sendo portanto, escassos os trabalhos de pesquisa acerca dos efeitos da implementação de gamificação aliada a Sistemas de Gestão de Aprendizagem, como por exemplo o Moodle e o Blackboard 9.

Piteira e Costa (2017) conduziram um estudo no sentido de fornecer uma proposta concreta de uma possível *framework* conceptual de gamificação aliada aos sistemas de gestão de aprendizagem, de modo a suportar a aprendizagem da programação em ambientes online.

Na conceção da *framework* proposta por Piteira e Costa (2017), foram definidos oitos principais componentes: (a) público-alvo, (b) objetivos gerais, resultados esperados e tópicos, (c) conteúdos, (d) princípios de desenho educacional gamificado, (e) mecânicas de jogo, (f) absorção cognitiva, (g) personalidade e (h) fluxo.

Em relação à primeira componente definida por Piteira e Costa (2017), podemos concluir que é importante a correta identificação do público-alvo de modo a ser possível estabelecer quais deverão ser os objetivos gerais, tópicos e resultados a esperar de um curso de aprendizagem de programação.

Na caracterização e estruturação de um curso é fundamental a identificação e definição prévia dos objetivos gerais e resultados esperados. A identificação dos tópicos a lecionar servem como suporte para que esses mesmos objetivos e resultados sejam alcançados. Piteira e Costa (2017), referem que existem diversas teorias que suportam este fato, dando especial enfâse à teoria de Bloom (1956). Esta teoria refere que existe um nível crescente de complexidade entre os diversos domínios cognitivos, onde os estudantes apenas devem adquirir uma competência cognitiva após terem dominado a competência que a antecede.

Segundo Piteira e Costa (2017), existem diversas tecnologias que suportam as estratégias educacionais quando nos referimos a plataformas online de aprendizagem. Nesse sentido, Oliver e Herrington (2003), constituíram uma *framework* que agrupava os diversos elementos tecnológicos que apoiavam essas estratégias acabando por identificar três principais áreas de aprendizagem: recursos, suportes e atividades. Ainda em relação aos conteúdos de um curso

orientado à aprendizagem de programação, Piteira e Costa (2017), efetuaram anteriormente diversos estudos acerca das dificuldades percecionadas pelos estudantes em relação aos diferentes tópicos abordados. Essas dificuldades deverão ser alvo de estratégias gamificadas orientadas à educação de modo a colmatar os problemas de envolvimento dos estudantes.

Dicheva e colegas (2015), definiram uma *framework* composta por dois principais componentes resultantes da proposta de definição de Gamificação realizada por Deterding e colegas (2011): (a) princípios de desenho educacional gamificado e (b) mecânicas de jogo. Na *framework* proposta por Piteira e Costa (2017), estes são dois dos seus principais componentes.

Para Piteira e Costa (2017), a componente dos princípios de desenho educacional gamificado é composta por diversos princípios que não são propriamente específicos dos desenhos dos jogos, uma vez que os mesmos têm sido amplamente utilizados noutras vertentes, como por exemplo, em sistemas instrucionais. A autora acrescenta ainda que a aplicação destes princípios nos tópicos e conteúdos de um curso, pode ser determinante em relação à interação dos estudantes com os diferentes componentes do curso, como meio de serem atingidos os resultados esperados identificados anteriormente. Estes princípios podem ser visíveis na tabela 2.2.

Tabela 2.2: Princípios de Desenho Educacionais (Adaptado de Piteira e Costa (2017))

Princípios de Desenho	**Descrição**
Progresso	Permite ao estudante visualizar a progressão no curso
Retorno	Retorno imediato após submissão das avaliações
Envolvimento Social	Inclui a cooperação e interação entre pares
Estado visível	Reputação, credibilidade social e reconhecimento dos resultados obtidos. Pressupõe que os resultados obtidos são visíveis a todos
Acesso bloqueado	Pressupõe que o conteúdo e atividades estão dependentes da realização e desempenho das anteriores
Liberdade para falhar	Pressupõe baixo risco na submissões de avaliação. São permitidas múltiplas tentativas
Restrição no tempo das submissões	Pressupõe que a submissão da avaliação, e.g. um quiz tem definido um tempo limite para ser submetido
Objetivos / Desafios	Definição de objetivos específicos, claros, de dificuldade moderada e imediatos
Customização	Definição de experiências personalizadas, dificuldade adaptada, desafios que são perfeitamente alcançáveis, aumentar a dificuldade à medida que as competências do estudante expandir

Narrativa	Pressupõe a apresentação do conteúdo programático e a definição das atividades enquadradas numa história que se desenrola ao longo do curso.
Liberdade de escolha	Inclui a possibilidade dos estudantes escolher: que tipo de desafios querem completar, e.g. contribuírem para um blog, completarem um quiz, criarem um vídeo educacional.
Surpresas, Prémios	Pressupõe a utilização de elementos surpresa e prémios.

A componente mecânicas de jogo proposta na *framework* de Piteira e Costa (2017), é compostas pelas mecânicas e dinâmicas de desenho de jogos. As mecânicas estão intimamente relacionadas com os elementos provenientes de jogos que podem ser adaptados, em contexto educativo, ao desenho e implementação de um curso, como por exemplo, medalhas, pontos, níveis e quadros de honra. Paralelamente, as dinâmicas estão relacionadas com o planeamento dos fluxos de interação que os que os estudantes deverão ter em relação aos diversos conteúdos e atividades do curso, ao longo do seu percurso.

Segundo Agarwal e Karahanna (2000), a absorção cognitiva pode ser definida como o estado de profundo envolvimento com o *software*, sendo constituída por cinco dimensões: (a) dissociação temporal, que pode ser traduzida como a incapacidade de ter a noção do tempo que se despendeu numa atividade, (b) imersão focada, ou a experiência de total envolvimento, onde tudo o que requer a atenção do utilizador é ignorado, (c) prazer acrescido, onde o utilizador têm uma maior capacidade de capturar os aspetos prazerosos da interação com o *software*, (d) controlo, representa a perceção que o utilizador têm de estar sobre controlo das suas interações com o *software*, e (f) curiosidade, que representa a curiosidade sensorial e cognitiva que a experiência da interação com o *software* lhe suscita. Seguindo esta definição, Piteira e Costa (2017), referem que a componente dos princípios de desenho educacional gamificado em conjunto com as mecânicas de jogo, têm a possibilidade de criar uma atmosfera envolvente, que poderá contribuir para uma maior absorção cognitiva por parte dos estudantes, em relação à aprendizagem da programação.

Em relação à componente da personalidade, Piteira e Costa (2017) verificaram que a mesma era composta por cinco dimensões: (a) extroversão, (b) amabilidade, (c) conscienciosidade, (d) estado emocional e (e) abertura a novas experiências. A personalidade do utilizador pode influênciar a sua absorção cognitiva e os seus resultados durante o período de aprendizagem da programação.

Por último, a componente fluxo identificada Piteira e Costa (2017), baseia-se no trabalho desenvolvido por Csikszentmihalyi (1990) que estuda o fato de os indivíduos ficarem tão envolvidos numa atividade sem a presença de incentivos externos. Csikszentmihalyi (1990) referiu que existe um estado de fluxo, ou fluxo de experiência, onde o individuo perceciona um balanço entre a dificuldade e ganho na concretização de uma atividade, sendo que essa perceção suscitará no individuo uma sensação de desafio e ao mesmo tempo confiança de que está sobre controlo. Assim, na *framework* proposta por Piteira e Costa (2017), esta componente terá impacto sobretudo na análise do estado de envolvimento dos estudantes em relação às suas experiências aquando os seus percursos em cursos de aprendizagem de programação gamificados.

Na figura 2.1, temos uma visão geral da *framework* proposta por Piteira e Costa (2017) onde podemos observar e analisar como todos os componentes identificados pela autora se relacionam.

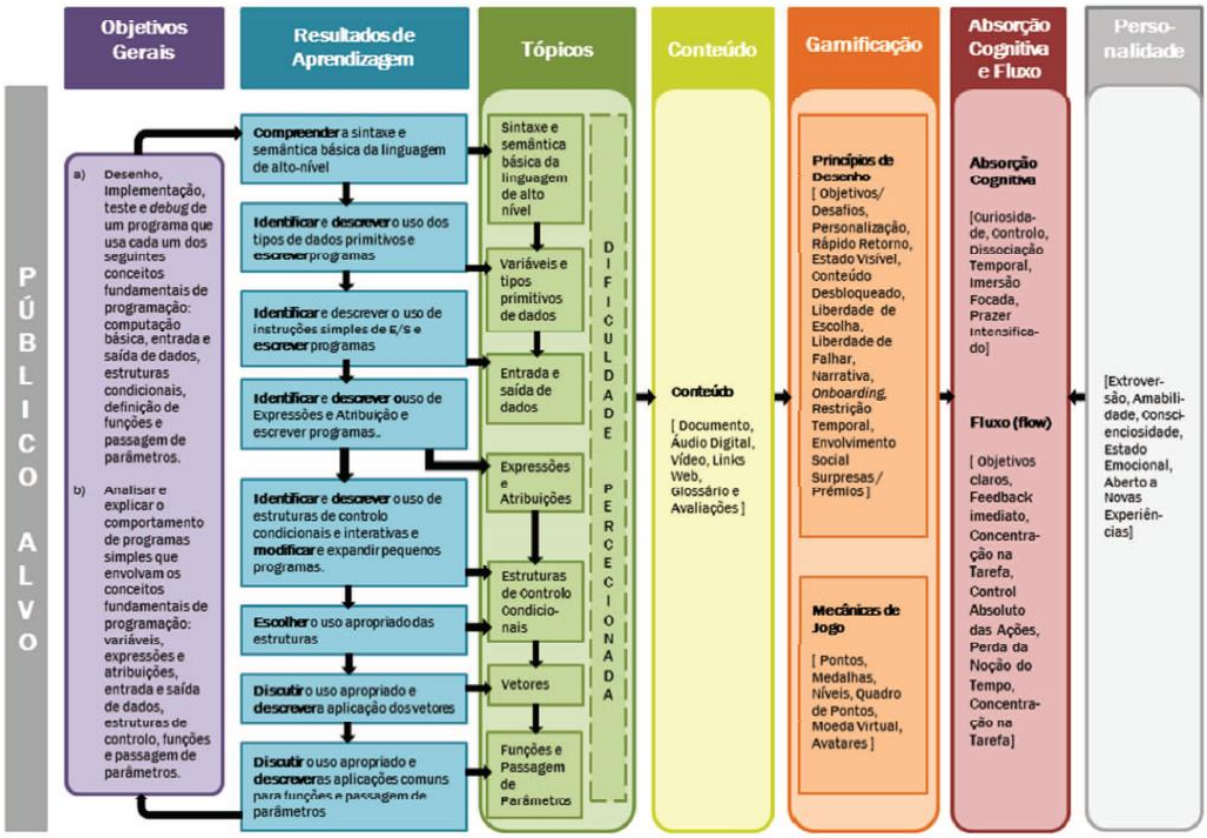


Figura 2.1: *Framework* teórica gamificada estendida (Piteira & Costa, 2017)

2.6 Sumário

Neste capítulo foi possível identificar quais os principais desafios aquando a aprendizagem da programação. Verificou-se que a falta de motivação e empenho por parte dos estudantes era o principal desafio a colmatar e que apesar das tentativas por parte de professores e investigadores em contrariar este facto, ainda não foi possível chegar a uma solução que efetivamente altere as motivações dos estudantes de programação.

Em seguida abordamos a gamificação explicando o seu conceito e em que medida a mesma poderia contribuir para aumentar a motivação e empenho dos estudantes perante a aprendizagem da programação.

Por fim, expusemos os moldes de uma *framework* conceptual gamificada que tem como intuito o aumento da motivação dos estudantes perante a utilização de sistemas de gestão de aprendizagem online que servirá de base para o desenvolvimento do modelo que iremos propor nesta dissertação. De referir que parte significativa a revisão da literatura aqui realizada foi já objeto de publicação em artigo de conferência (Pereira, Costa and Aparicio. 2017).

Neste capitulo foi realizada a revisão de literatura com o objectivo de melhor adequação e identificação das soluções que contribuem para a resolução do problema. Este capitulo 2 constitui as duas primeiras fases da abordagem metodológica.

3. Modelo Conceptual Gamificado

Neste capítulo iremos propor um modelo conceptual gamificado que servirá de suporte à implementação do protótipo que será posteriormente utilizado para a obtenção dos resultados. Este capitulo corresponde a terceira fase da abordagem metodológica da dissertação: Desenvolvimento e construção de um artefacto. Começa-se por detalhar as componentes constituintes do modelo e os elementos que os integram, bem como as suas relações. Por fim, serão apresentados os seus diagramas de caso de uso.

3.1 Definição do Modelo Conceptual Gamificado Proposto

De acordo com as informações obtidas aquando a realização do estado de arte, e no seguimento do trabalho realizado pelos autores que foram analisados anteriormente, foi possível identificar os diversos componentes que representam na íntegra um modelo gamificado que poderá servir de suporte à aprendizagem da programação.

No início da conceção deste modelo, verificou-se que seria importante perceber qual o impacto que a gamificação teria em relação ao aumento de utilização de um sistema de gestão aprendizagem de programação. Tendo em conta esta necessidade, propomos um modelo no qual existam duas vertentes distintas que deverão ser alvo de análise e comparação, para se conseguir chegar a uma conclusão acerca da hipótese inicialmente proposta: a vertente gamificada e a vertente não gamificada.

O modelo conceptual que foi desenvolvido aquando a realização desta dissertação, pode ser visto como um curso flash de introdução à programação, onde os estudantes teriam apenas um dia para completar todos os objetivos propostos pelo modelo, bem como alcançar os seus objetivos pessoais.

Neste modelo podemos verificar a existência de cinco principais componentes: (a) Objetivos Gerais do Curso, (b) Conteúdos Programáticos, (c) Conteúdos Gerais, (d) Conteúdos Gamificados e (e) Gamificação. Estes dois últimos componentes apenas estariam presentes na vertente gamificada.

Uma vez que foram identificados inúmeros desafios em relação à aprendizagem da programação foi importante estipular corretamente objetivos que os colmatassem. São então

propostos os seguintes objetivos: (a) Aumentar a participação dos estudantes, (b) Aumentar a motivação dos estudantes, (c) Aumentar a curiosidade dos estudantes pela programação em geral, (d) Alargar o conhecimento dos estudantes acerca de uma linguagem de programação específica, (e) promover a aprendizagem autodidata e (f) facilitar o processo de avaliação.

Ao nível dos conteúdos programáticos, existiu uma grande preocupação em efetuar um planeamento prévio de modo a que os mesmos não provocassem um efeito negativo nos estudantes, gerando estados de ansiedade e confusão. Decidiu-se que seria ideal integrar conceitos básicos de uma linguagem de programação, uma vez que apenas deste modo seria possível que os estudantes conseguissem desenvolver mentalmente as bases de programação.

O quadro de conteúdos programáticos presentes no modelo proposto, tanto na vertente gamificada, como na vertente não gamificada, é o mesmo, e é composto por três principais categorias: (a) variáveis e programação condicional, (b) ciclos e (c) *arrays*.

É importante que os estudantes consigam ter uma visão geral de conceitos básicos como as variáveis primitivas e as suas utilizações no universo da programação. Propomos portanto, que se comece com uma abordagem introdutória às variáveis primitivas e às suas aplicações em relação à programação condicional.

Após os estudantes terem conseguido absorver cognitivamente estes simples conceitos da programação, propomos que sejam abordados tópicos relacionados com a programação cíclica como segunda competência a ser adquirida.

Por fim, o último tópico que deve ser abordado, e o mais complexo, devem ser os *arrays*. Durante o contacto com este tópico académico, os estudantes devem (a) aprender a declarar e a atribuir valores a um *array*, (b) aceder aos seus elementos e a (c) percorrer um *array* através dos ciclos anteriormente mencionados. Achamos também que seria importante abordar algumas das propriedades e métodos mais utilizados e importantes de um *array*, e portanto, propomos uma abordagem à propriedade *length* de um *array* e aos métodos responsáveis pela adição e remoção de elementos de um *array*.

Achamos que estes seriam os conteúdos programáticos mais importantes a abordar num curso de introdução à programação, pelo que é importante o fornecimento de exemplos ao longo de todo o contacto com estes tópicos uma vez que estas matérias podem ter um impacto crucial na compreensão e desenvolvimento de algoritmos mais complexos por parte dos estudantes.

Em relação aos conteúdos gerais do modelo, em ambas as versões, os estudantes deverão ter a possibilidade de realizar desafios acerca dos conteúdos programáticos referidos anteriormente, bem como visualizar as suas estatísticas e concretizações através de uma página que deve ser desenvolvida para esse efeito.

Propomos que todos os desafios comecem com uma descrição teórica do que vai ser abordado. Devem ser fornecidas breves explicações teóricas e apresentados exemplos das aplicações possíveis dos conteúdos abordados no mundo da programação. Os estudantes apenas deverão começar o desafio propriamente dito quando todos os conteúdos programáticos tiverem sido absorvidos cognitivamente. Deste modo, os estudantes terão a oportunidade de investigar mais sobre os tópicos abordados, desenvolvendo as suas capacidades de aprendizagem autodidata, fator extremamente importante na aprendizagem de qualquer linguagem de programação.

Aquando a realização dos desafios propomos que sejam apresentadas aos estudantes três questões em formato *quiz*, onde deverão ser analisados exemplos práticos de programação e identificados os *outputs* correspondentes. Desta forma os estudantes conseguirão ter uma perceção acerca dos processos de programação envolvidos na linguagem que está a ser lecionada. Após terem sido respondidas as três questões mencionadas, os estudantes deverão receber um desafio final ao estilo "tente você". Neste desafio, os estudantes deverão analisar um bloco de código e completá-lo de acordo com o que for pedido numa questão que se encontrará subjacente. Com este tipo de exercício, os estudantes terão um maior contato com a programação, o que poderá promover a curiosidade e atração pela mesma.

Uma vez que a gamificação se preocupa com o fornecimento de feedback instantâneo, após a concretização dos desafios deverão ser expostos os resultados obtidos através dos mesmos. Com esta metodologia, os estudantes saberão em que patamar se encontram e se os seus conhecimentos foram interiorizados.

Propomos a existência de uma página onde os estudantes possam visualizar as suas estatísticas e concretizações. Esta página é crucial uma vez que transmite aos estudantes o sentimento de concretização e de recompensa pelo esforço empreendido aquando a aprendizagem da programação bem como fornece o feedback necessário para que o mesmo saiba se está a evoluir ou não. Esta página deve também servir o propósito de facilitar o processo de avaliação dos estudantes, dado que os mesmos passarão a saber à partida quais serão as suas notas finais.

Para que a gamificação possa surtir um efeito positivo ao nível da motivação dos estudantes, as tarefas devem ser ordenadas de acordo com o seu grau de dificuldade e complexidade. Portanto, devemos focar-nos no estudo das bases da programação e consequentemente aumentar o grau de dificuldade da matéria lecionada consoante os estudantes adquiram mais competências e aptidões. Assim estaremos a ir ao encontro da teoria de flow proposta por Csikszentmihalyi (1990), onde os estudantes não se sentirão demasiado desafiados a fim de perderem o interesse e a motivação, ao mesmo tempo que serão proporcionados desafios curtos e concretizáveis, tal como sugere a gamificação. Por fim, é importante realçar que os conteúdos lecionados devem ser enquadrados com um perfil de estudante onde os conhecimentos base da programação sejam escassos ou inexistentes.

É importante a existência de uma *interface* gráfica bastante viva, apelativa e motivadora, tal como acontece com a maior parte dos jogos em dispositivos digitais modernos, para que os estudantes possam ter uma experiência agradável de modo a despertar a curiosidade e participação. Esta *interface* deve ser o mais intuitiva possível para que o sentimento de obrigação em ir ao encontro de objetivos definidos e definitivos seja inexistente. Cada estudante deverá ter a possibilidade de navegar conforme entender, fomentando assim um sentimento de exploração característico da gamificação (Rodrigues et al., 2014; Rodrigues et al., 2016 a; 2016 b).

Apesar do fato anteriormente descrito, o modelo proposto deverá seguir a vertente Ludus, especialmente na vertente gamificada, uma vez que as ações existentes são definitivas e não existe muito espaço para que os estudantes possam fazer as suas próprias regras e definir os seus objetivos.

Uma vez que a gamificação se preocupa com o (a) aumento da participação por parte dos estudantes , (b) promoção da aprendizagem autodidata , (c) tornar o processo de avaliação mais fácil e eficaz tanto para o estudante como para o professor e com o (d) fornecimento de feedback instantâneo, decidimos que seria importante ter especial atenção às mecânicas de jogo e aos padrões de desenho de jogo.

Na vertente gamificada deverão ser adicionados (a) pontos, (b) níveis, (c) crachás e (d) uma tabela classificativa. Estes elementos compõe as mecânicas de jogo e servem de suporte ao aumento da participação e da motivação intrínseca dos estudantes através da utilização de motivadores extrínsecos. Em relação aos padrões de desenho de jogo, devem estar presentes os

seguintes elementos: (a) restrições temporais, (b) narrativa cativante, (c) barras de progresso, (d) envolvimento social, (e) existência de desafios concretizáveis, (f) fornecimento de feedback instantâneo e (g) liberdade para falhar. Estas mecânicas e padrões de desenho de jogo devem ser definidas no sentido de colmatar os principais desafios aquando o ensino da programação. Em contraste, nenhum dos elementos referidos devem ser implementado na vertente não gamificada.

Após a definição dos elementos de gamificação que deveriam estar presentes na vertente gamificada, verificámos que seria importante alterar os conteúdos gerais de modo a criar uma atmosfera envolvente semelhante a um jogo.

Propomos que a página relativa aos desafios seja o mais gamificada possível uma vez que esta página será o foco principal dos estudantes a fim de concretizarem os seus objetivos pessoais bem com os propostos pelo protótipo.

Tendo em conta a componente das restrições temporais, propomos a adição de um temporizador a começar do zero de modo a que hajam penalizações consoante o tempo despendido na resolução dos *quizzes* e desafio final. O temporizador deverá começar do zero e não a partir de um valor fixo, uma vez que não é desejável a imposição de pressão acrescida aos estudantes, mas sim, a possibilidade de exploração autodidata dos conteúdos a fim de alcançarem o objetivo máximo de absorverem cognitivamente os tópicos abordados.

Em relação ao feedback relativo ao número de respostas certas e erradas, propomos que sejam utilizadas barras de progresso na vertente gamificada em detrimento de simples números informativos que apenas seriam implementados na vertente não gamificada. Os estudantes deverão começar com uma barra de vida totalmente preenchida correspondente ao número de questões por responder, que diminuiria consoante as respostas incorretas, bem como uma barra de experiência que começaria vazia e que iria aumentando consoante as respostas acertadas. Assim, iria haver uma maior perceção acerca de como o desafio estaria a decorrer, uma vez que a forte componente visual das barras de progresso não passaria despercebida.

É também importante dar ênfase às conquistas alcançadas em cada desafio relativamente a crachás, e como tal propomos que exista um mecanismo que forneça esse feedback após a realização do desafio. Desta forma os estudantes seriam recompensados de acordo com as suas conquistas pessoais ao mesmo tempo que iria haver a promoção do aumento das suas motivações perante a conquista dos restantes crachás.

Para além da divulgação dos crachás conquistados, no final de cada desafio, tanto na vertente gamificada como na vertente não gamificada, deverá existir um mecanismo que forneça aos estudantes o feedback relativo aos resultados obtidos no final do desafio. Na vertente gamificada, esses resultados deverão ser mais completos e complexos uma vez que serão utilizados pontos em vez de valores, e serão adicionados campos de avaliação extra correspondentes às restrições temporais, bem como ao uso ou não de dicas, que serão na sua essência as descrições teóricas e exemplos fornecidos mencionados anteriormente. Ainda na vertente gamificada, deve ser analisado o número total de pontos dos estudantes de modo a saber se houve uma subida de nível.

Outra funcionalidade que também deve ser implementada é a possibilidade de rever, alterar e submeter novamente o código do desafio final após término do desafio global. Esta funcionalidade deve ser implementada em ambas as vertentes uma vez que é desejável o fornecimento da possibilidade aos estudantes de reverem os seus erros e procurarem uma nova solução de modo a evoluírem a partir dos mesmos.

Na página de estatísticas, deve haver um mecanismo que permita realçar as conquistas relativas aos crachás para além dos dados referentes às suas conquistas pessoais. Assim estaremos a fomentar o espírito de aventura e a despertar o espírito de competição presente em cada estudante que passará a sentir-se mais motivado e empenhado em aprender a programar de modo a colecionar todos os crachás.

Na vertente gamificada deve existir um *widget* que indique a situação atual dos estudantes para que os mesmos consigam ter sempre a perceção de qual o seu estado atual perante o curso. Através deste *widget* deverá ser possível visualizar as informações acerca da pontuação total e nível atual dos estudantes.

É imperativa a promoção da competitividade e do envolvimento social, e portanto deverá existir uma página de classificação geral, onde os estudantes possam observar, através de uma tabela classificativa, as suas conquistas bem como as dos seus colegas. Desta forma os estudantes saberão em que situação se encontram perante o quadro geral do curso.

Na seguinte figura é possível obter uma visão geral sobre o modelo conceptual desenvolvido de acordo com os princípios da gamificação explorados anteriormente.

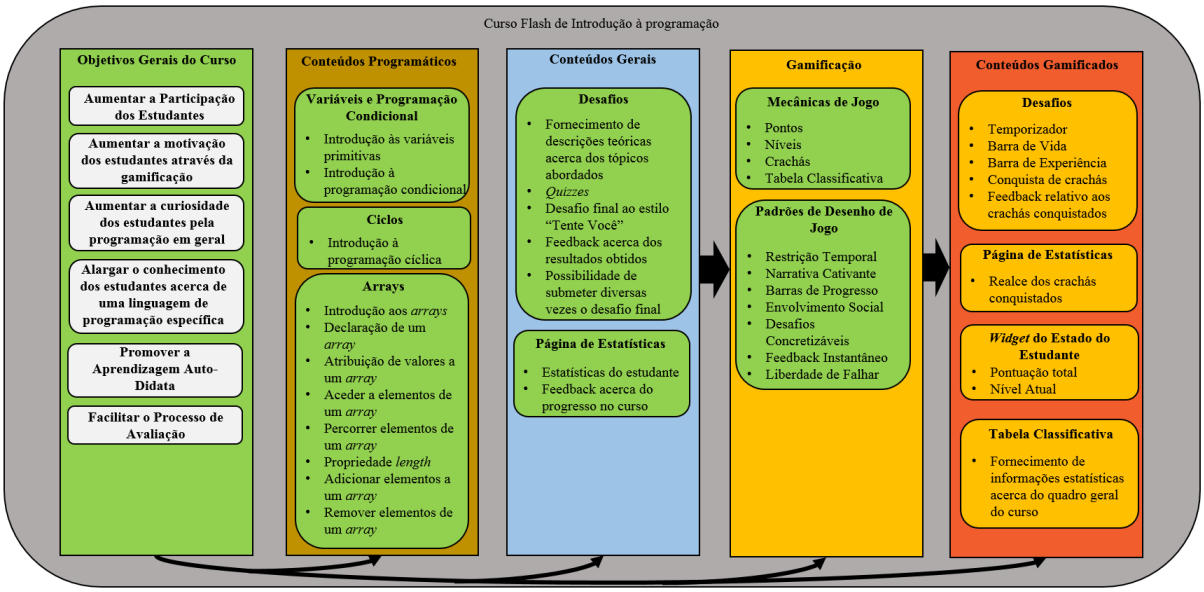


Figura 3.1: Modelo Conceptual Gamificado

3.2 Diagramas de Caso de Uso

Em seguida iremos descrever os casos de uso existentes no modelo conceptual que foi proposto. Para se obter uma melhor perceção das diferenças entre a vertente não gamificada e a vertente gamificada serão apresentados diagramas de casos de uso correspondentes às diferentes páginas que deverão estar presente em qualquer implementação do modelo: (a) página inicial, (b) página de seleção dos desafios, (c) página de desafio, (d) página das estatísticas e (e) página da tabela classificativa.

Na página inicial os estudantes deverão ter a possibilidade de seguir diversos caminhos que o guiarão para outras páginas com propósitos específicos. Tal como é possível verificar através da figura 3.2, na vertente gamificada para além da possibilidade dos estudantes irem para a página de seleção dos desafios e para página das estatísticas, deve ainda existir a possibilidade dos estudantes irem para uma terceira página correspondente à página da tabela classificativa. Para além deste fato é importante referir que esta página na vertente gamificada deverá ter um *design* gamificado de modo a que a mesma se assemelhe o mais possível a um jogo e também deverá existir um *widget* que esteja presente tanto nesta página como em todas as outras anteriormente referidas de modo a que os estudantes possam ter sempre a perceção da sua situação atual relativamente às suas pontuações e níveis atuais.

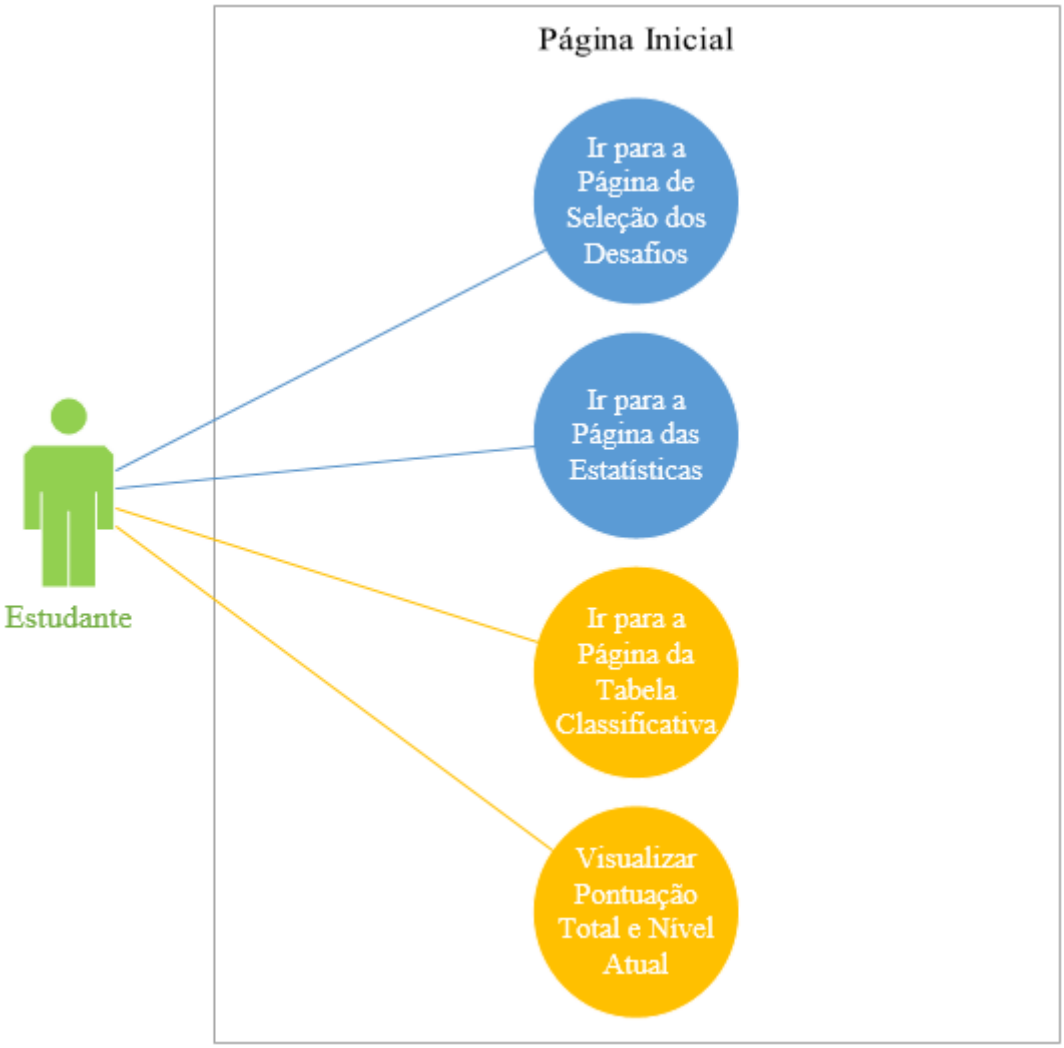


Figura 3.2: Página Inicial

Como é possível verificar através da figura 3.3, correspondente à página de seleção dos desafios, não devem existir diferenças a nível funcional mas sim a nível de *design* sendo que na vertente gamificada deverá existir uma interface gráfica mais dinâmica e robusta tal como sugere a gamificação de modo a criar um ambiente envolvente que permita a existência de uma narrativa cativante para que os estudantes se sintam mais motivados e empenhados em aceitar os desafios propostos.

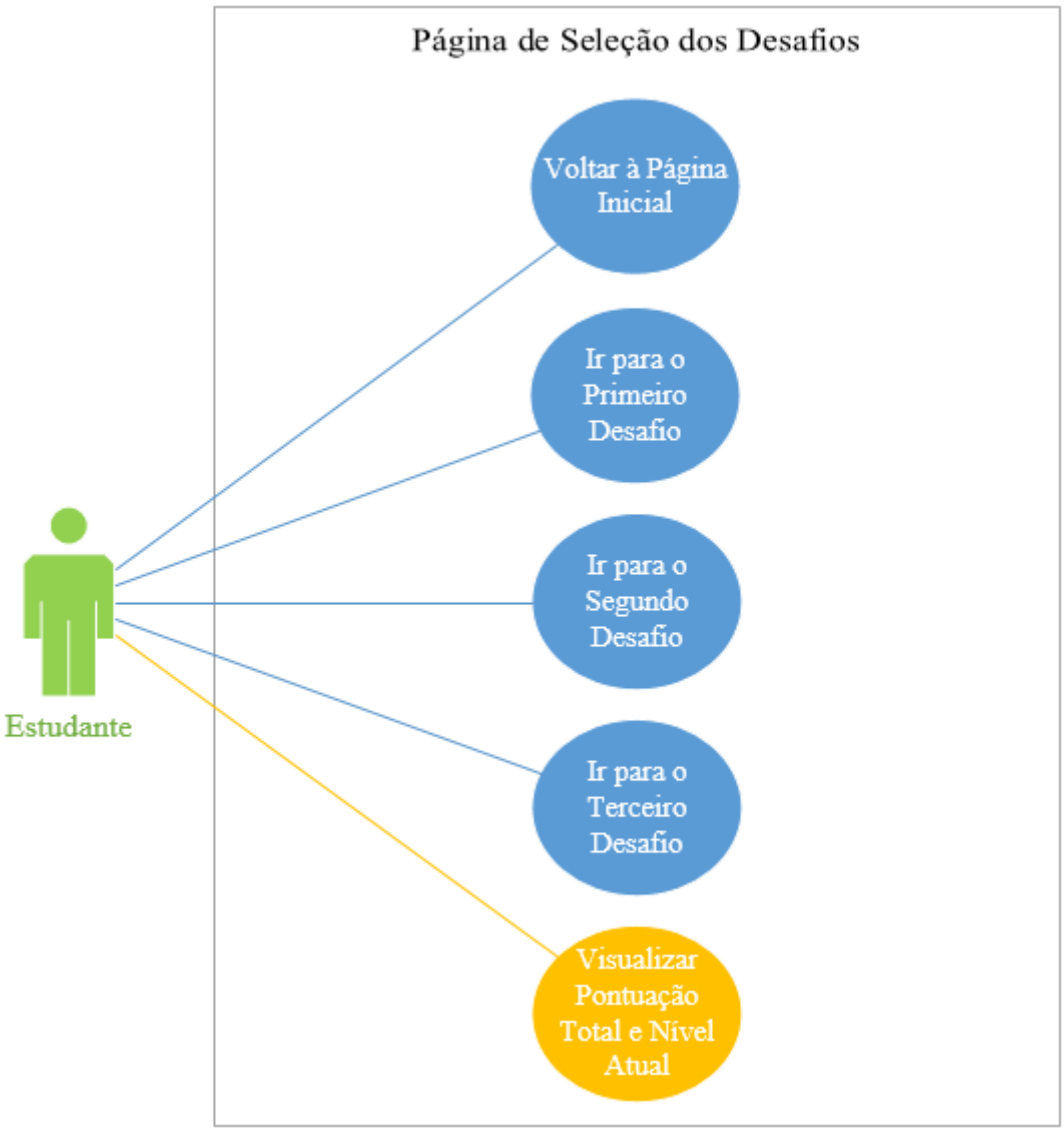


Figura 3.3: Página de Seleção de Desafios

A página de desafio representada pela figura 3.4, tal como foi mencionado anteriormente, deverá ser o maior foco de implementação de gamificação. Portanto, não deverão existir grandes diferenças a nível funcional, mas a nível de *design* na vertente gamificada, deverão ser adicionados (a) um temporizador, (b) uma barra de vida, (c) uma barra de experiência, (d) existência de crachás a serem conquistados e (e) um mecanismo que forneça o feedback relativo à conquista dos crachás obtidos aquando a concretização do desafio bem como as restantes pontuações obtidas durante o desafio. Desta forma estaremos a promover uma experiência jogável para os estudantes.

Figura 3.4: Página de Desafio

Em relação ao diagrama de casos de uso da página das estatísticas presente na figura 3.5, as grandes diferenças das duas vertentes existentes no modelo proposto deverá incidir sobretudo

na quantidade de informação a apresentar aos estudantes bem como na existência de um mecanismo que destaque as conquistas dos crachás na vertente gamificada.

Figura 3.5: Página das Estatísticas

Por fim, a página tabela classificativa terá como propósito máximo o fornecimento de todos os detalhes necessários para que os estudantes saibam em que posição se encontram tendo em conta as conquistas dos seus colegas do curso.

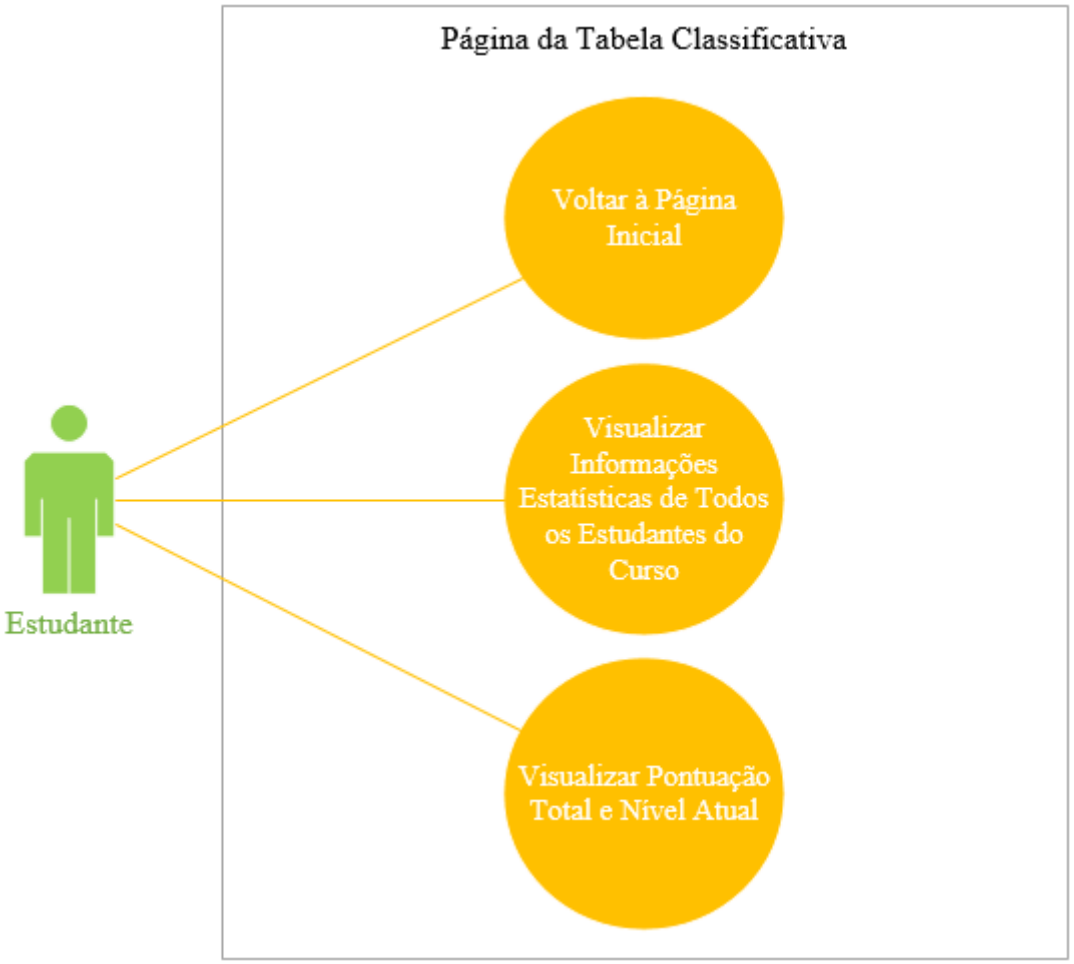


Figura 3.6: Página da Tabela Classificativa

4. Implementação

Neste capítulo iremos descrever detalhadamente todos os aspetos relevantes acerca do desenvolvimento e implementação do protótipo proposto como solução para os problemas da aprendizagem da programação, "Kaboom Academy". Os tópicos abordados serão os requisitos do sistema, iremos fazer uma visão geral do protótipo e por fim falaremos acerca da arquitetura do sistema. Este capítulo corresponde a quarta fase da abordagem metodológica: Protótipo e uso do artefacto.

4.1 Requisitos do Sistema

Para que qualquer aplicação tenha sucesso, é necessário efetuar uma análise de requisitos previamente à sua conceção e desenvolvimento. Estes requisitos estão agrupados em duas categorias, os requisitos funcionais e os requisitos não funcionais. Os requisitos funcionais descrevem as funcionalidades do sistema bem como os comportamentos expectáveis aquando a sua utilização, enquanto os requisitos não funcionais descrevem a arquitetura do sistema.

Da solução conceptual proposta derivaram os seguintes requisitos funcionais:

- Gamificação – Este é o principal requisito, que terá obrigatoriamente de estar presente, de modo a obtermos respostas concretas em relação ao problema inicialmente identificado. Com este requisito, o estudante irá sentir-se mais motivado para usar o protótipo e explorar os seus conteúdos.

- Interface do utilizador – Este requisito é necessário para que o estudante consiga usufruir ao máximo da experiência de utilização e, portanto, especificámos páginas minuciosamente para que fosse possível uma experiência mais agradável.

- Sistema de autenticação – Para que seja possível definir e atualizar as informações de cada um dos utilizadores do protótipo, ficou definido que iria ser implementado um sistema de login. Para o efeito será utilizado o serviço de autenticação do Firebase. De maneira a retirar alguma complexidade no desenvolvimento do protótipo, foi implementado um login usando contas Google, pelo que o estudante deverá ser detentor de uma a fim de utilizar o protótipo.

- Armazenamento e processamento de dados – A maioria das aplicações deverá ter uma maneira de armazenar e processar os dados relativos aos seus utilizadores, geralmente através de uma base de dados. Optámos por utilizar o serviço de base dados do Firebase, pela sua fácil e rápida implementação, e por estar orientado a uma gestão dos dados em tempo-real.

- Estatísticas do utilizador – Uma vez que o protótipo implementado foi desenvolvido para pessoas que pretendam aprender a programar, foi definido que seria crucial haver uma página onde os estudantes conseguissem visualizar os seus progressos de desenvolvimento.

- Conteúdos educacionais de programação em JavaScript – Para conseguirmos obter um feedback acerca das motivações dos estudantes perante a aprendizagem da programação, foi necessário definir que existiria uma secção onde os mesmos poderiam aprender e testar os seus conhecimentos acerca de uma das linguagens de programação mais utilizadas e acessíveis de aprender, o JavaScript.

Em relação aos requisitos não-funcionais foram definidos os seguintes:

- Acessibilidade – Em relação à acessibilidade, e dadas as tecnologias utilizadas no desenvolvimento do protótipo, ficou definido que o mesmo deverá ser um *website* capaz de ser executado nos exploradores Google Chrome, Windows Edge e Mozilla Firefox. Uma vez que não se trata de uma aplicação híbrida, o protótipo apenas foi contemplado e testado nos exploradores anteriormente referidos. No entanto, uma vez que foi utilizado Twitter Bootstrap para definição de estilos, este protótipo também funciona nos exploradores mencionados em dispositivos móveis.

- Usabilidade – Todas as páginas foram desenhadas de modo a que seja intuitiva a sua utilização por parte do estudante, sem necessidade de qualquer tipo de treino ou ensinamento prévio.

- Eficiência – Uma vez que foi utilizada a versão Spark do Firebase, que é essencialmente a versão free, o protótipo permitirá que até cem utilizadores estejam conectados à base de dados em simultâneo. Outro ponto importante a salientar, é o facto de ser necessário uma conexão à internet. A atualização de novos dados dos estudantes depende dessa conexão, mas caso a mesma esteja com intermitências é garantido que os dados não serão perdidos.

- Confiabilidade – Após ser realizado o *deploy* do protótipo para a internet através do serviço de hospedagem do Firebase, é garantido que a sua utilização estará sempre disponível a qualquer altura do dia. Em relação a possíveis falhas no armazenamento dos dados por inconsistências na estrutura dos mesmos, foram definidos modelos dos objetos existentes na base de dados de modo a mitigar o risco de problemas futuros.

4.2 Visão Geral do Protótipo

O protótipo desenvolvido está dividido em diversos módulos que serão descritos na secção posterior, sendo que as páginas a que o estudante irá ter acesso estão agrupados no módulo core que se encontra subdividido em duas principais categorias, a área pública e a área privada.

4.2.1 Área Pública

Na figura 4.1 podemos observar a principal e única página da área pública, a página de login.

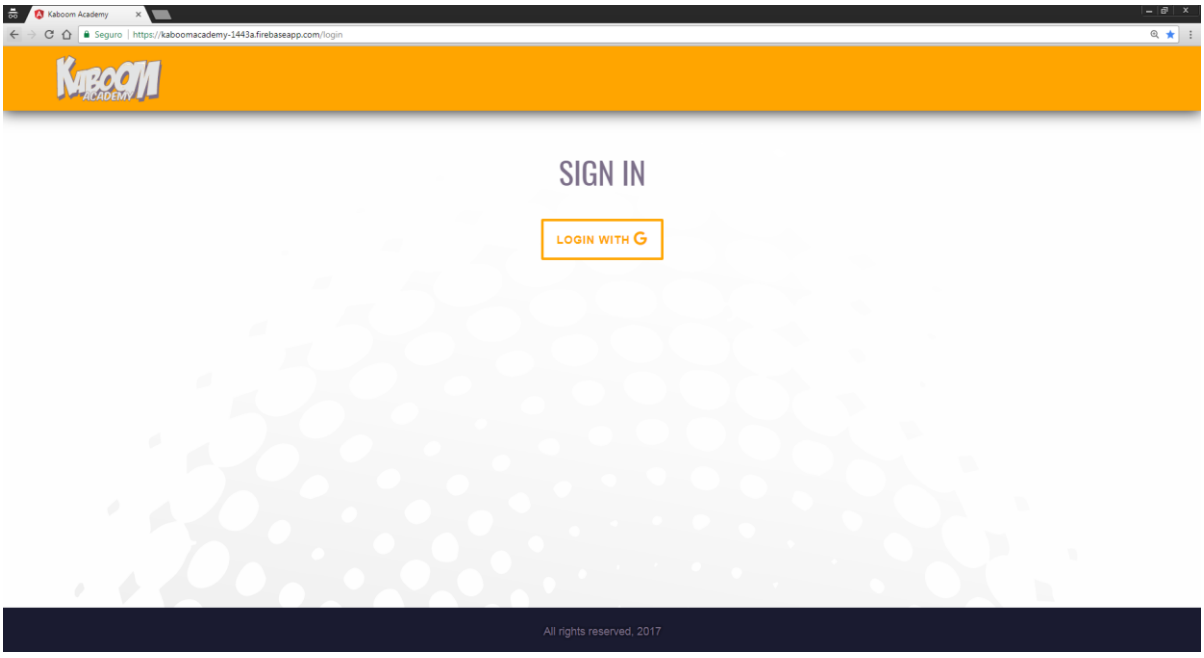


Figura 4.1: Página de Login

Através desta página será possível efetuar o login que dará acesso às páginas da área privada que serão analisadas posteriormente. Como foi referido anteriormente, esta página serve o propósito de cumprir um dos requisitos funcionais, onde os estudantes deverão autenticar-se.

Assim que se tente efetuar o login irá surgir uma pop-up com a solicitação dos dados de uma conta Google que o estudante possua e deseje inserir como subscrição ao protótipo Kaboom Academy.

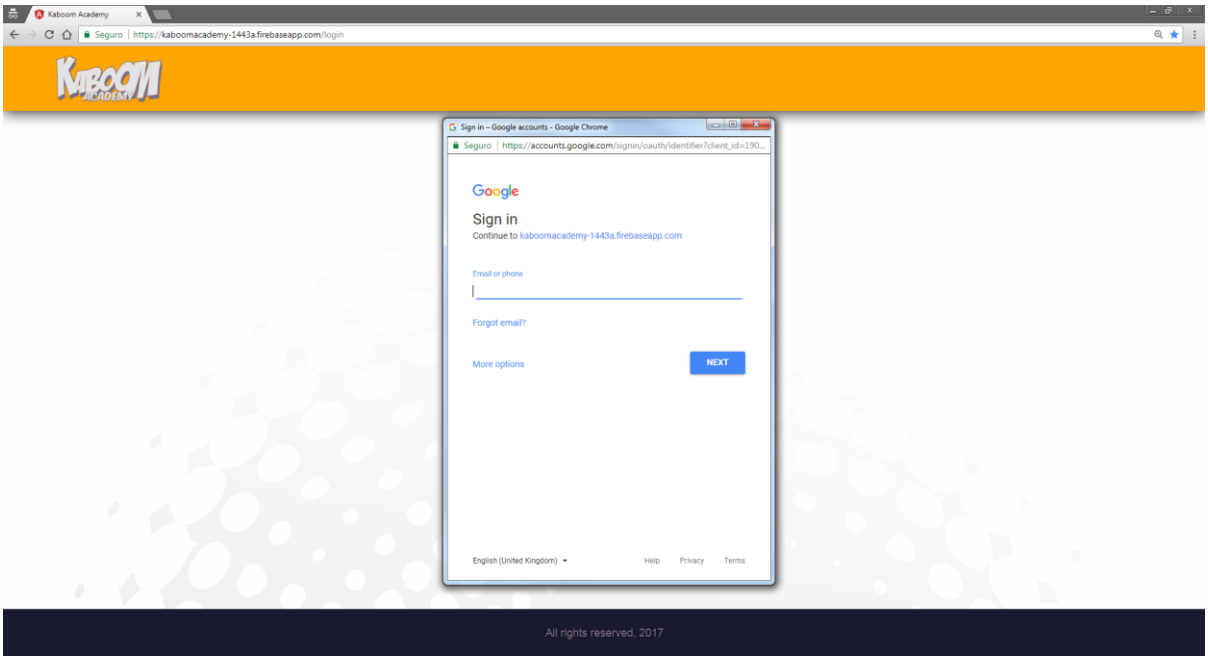


Figura 4.2: Pop-up de Autenticação

Caso o estudante insira com sucesso uma conta Google válida, o mesmo será remetido para a página inicial, um dos principais ecrãs da área privada. Caso o estudante falhe a autenticação, ou tenha havido um problema a autenticar por questões alheias ao estudante, como por exemplo falhas na internet, irá surgir uma mensagem proveniente dos servidores do Firebase com a indicação do tipo de erro ocorrido, como podemos verificar na figura 4.3.

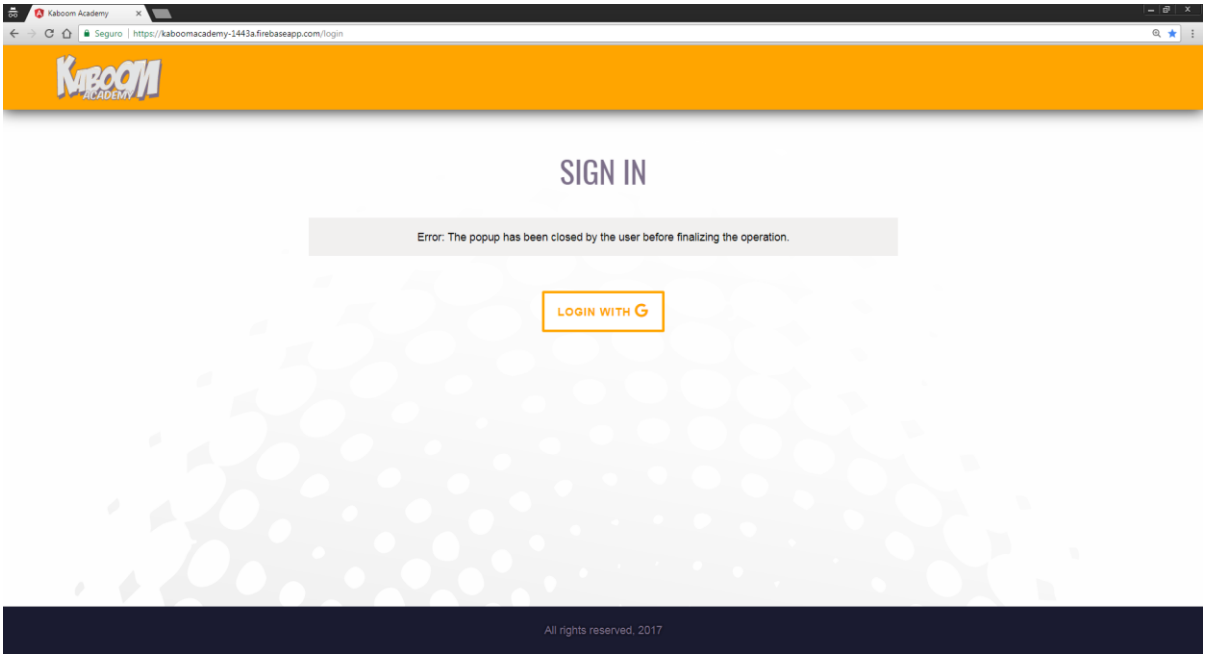


Figura 4.3: Erro ao Autenticar

4.2.2 Área Privada

A área privada é composta por diversas páginas distribuídas em dois modos distintos, o modo gamificado e o modo não gamificado, sendo que na secção 4.3 irá ser explicada a atribuição deste modo.

Em ambos os modos existe uma página inicial, uma página de seleção de desafios, uma página de desafio e uma página de estatísticas do utilizador. No modo gamificado temos ainda uma página adicional onde o estudante poderá visualizar a tabela classificativa.

Apesar das páginas serem semelhantes a nível funcional, os seus nomes serão diferentes consoante o modo em que o estudante estiver inserido.

4.2.2.1 Página Inicial

Na figura 4.4 e 4.5, podemos observar os ecrãs da página inicial no modo não gamificado e no modo gamificado, respetivamente.

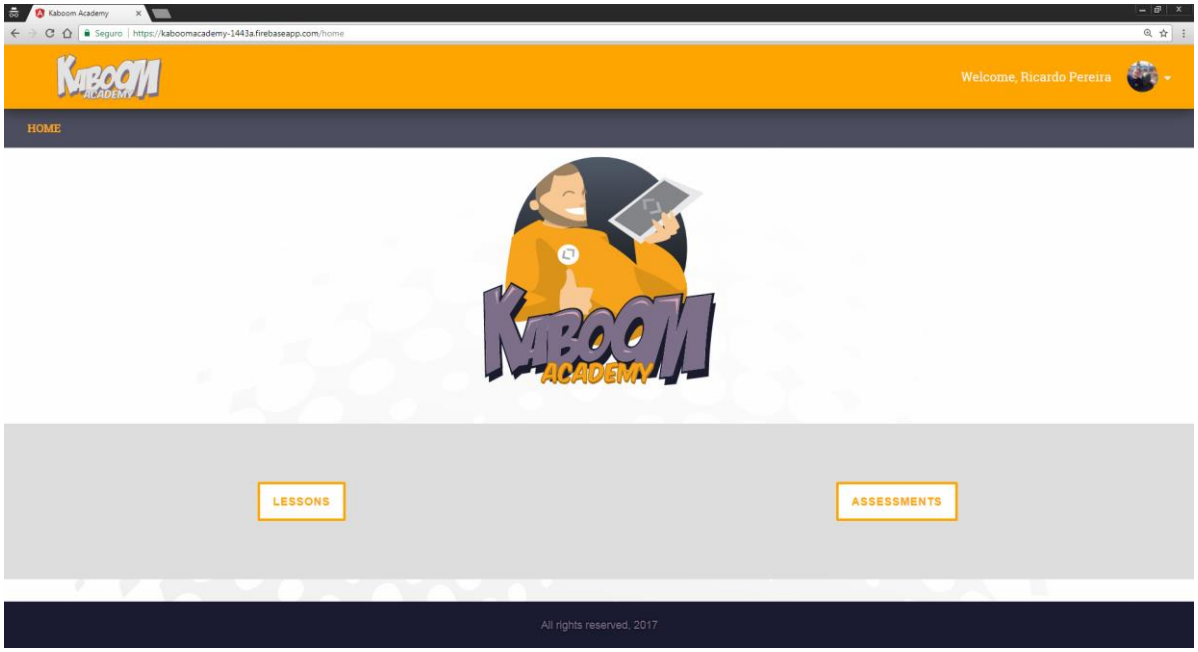


Figura 4.4: Página Inicial em Modo Não Gamificado

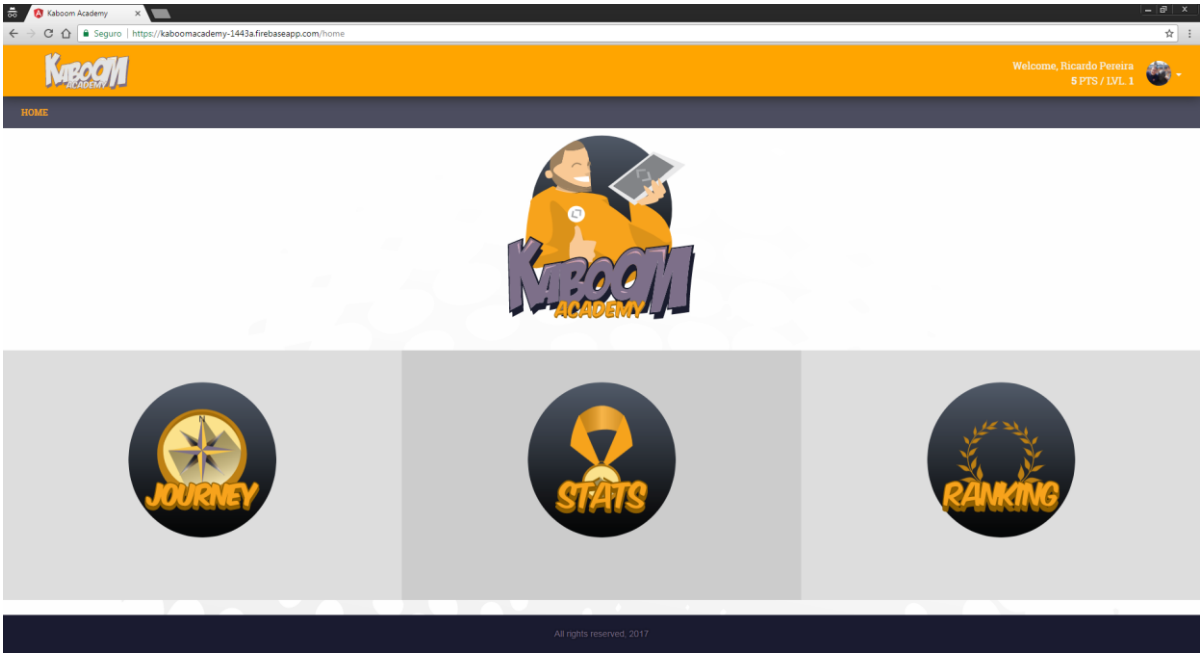


Figura 4.5: Página Inicial em Modo Gamificado

Conseguimos à partida identificar duas grandes diferenças da versão não gamificada para a versão gamificada da página inicial (a) a existência de um *widget* no lado direito do cabeçalho do *website* com a indicação da pontuação e nível do estudante e (b) o design característico de um jogo na seleção das páginas em vez de simples botões. Desta forma estaremos a promover o despertar da curiosidade e consequentemente o aumento da motivação e participação dos estudantes perante a utilização do protótipo.

A página inicial serve como ponto de partida, e através dela o estudante terá a possibilidade de aceder às restantes áreas do protótipo:

- Página "Minha Jornada" no modo gamificado ou página "Minhas Lições" no caso de o estudante estar no modo não gamificado.
- Página "Minhas Estatísticas" no modo gamificado ou página "Minhas Avaliações" no modo não gamificado.
- Página tabela classificativa, disponível apenas no modo gamificado.

4.2.2.2 Página "Minhas Lições" e Página "Minha Jornada"

Através da página "Minhas Lições", o estudante irá ter a opção de selecionar a lição que pretenda através da escolha de uma entre três categorias de lições existentes: "Variáveis e Programação Condicional", "Ciclos" ou "Arrays", como podemos visualizar na figura 4.6.

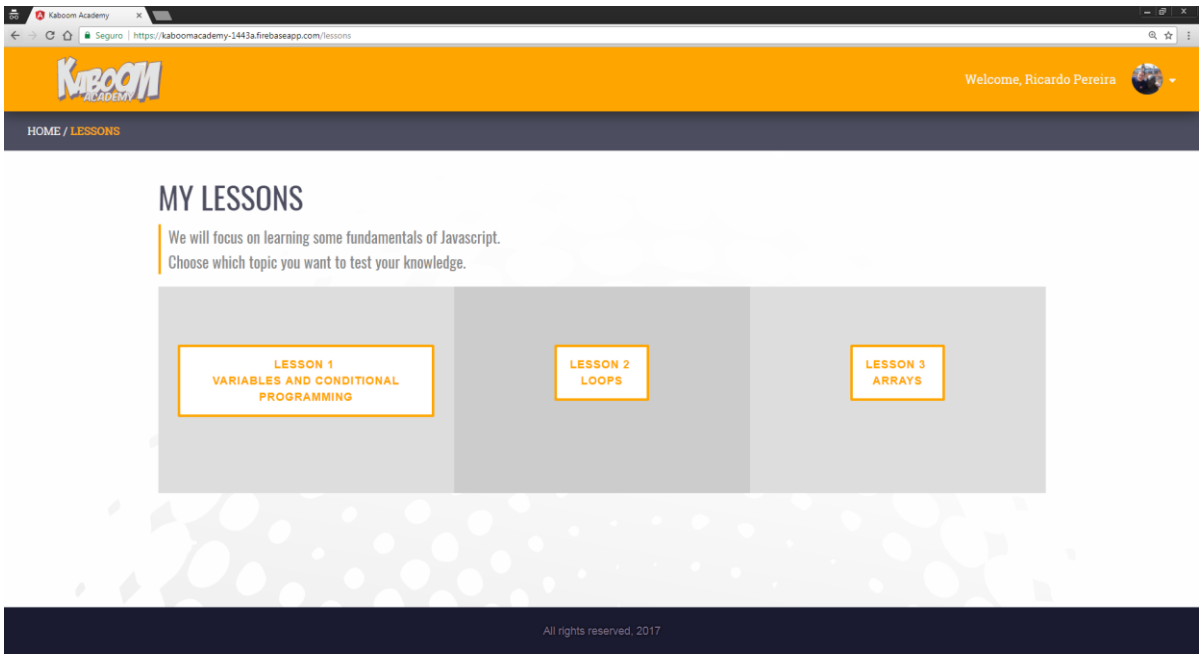


Figura 4.6: Página "Minhas Lições"

A página "Minha Jornada" têm as mesmas funcionalidades que a página "Minhas Lições", mas é ligeiramente diferente a nível visual, tal como observado na figura 4.7. Este facto prende-se com a implementação de gamificação onde optámos por implementar uma narrativa baseada na conquista de castelos, onde cada castelo corresponderá às categorias mencionadas anteriormente. Esta narrativa é implícita mas ao mesmo tempo muito intuitiva uma vez que

cada castelo têm um número associado à semelhança de um nível de um jogo. O segundo e terceiro castelos começarão bloqueados e apenas após a conquista do castelo que os anteceda será possível tentar conquistá-los.

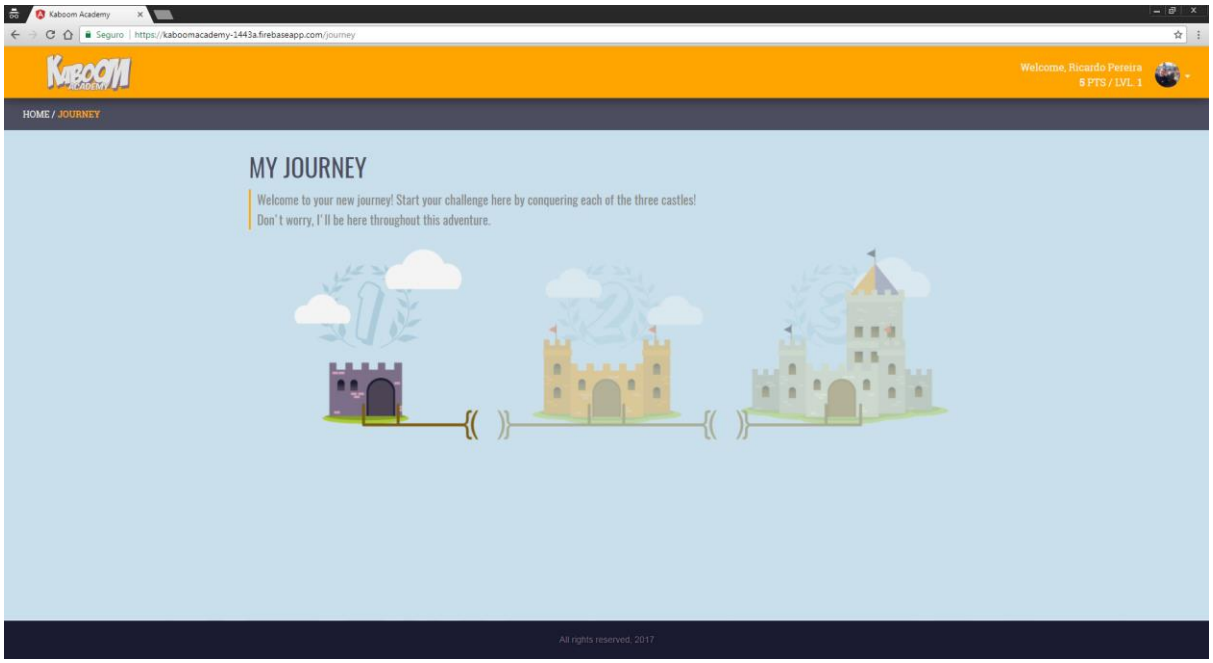


Figura 4.7: Página "Minha Jornada"

4.2.2.3 Página "Minha Lição" e Página "Minha Busca"

Esta é uma das principais páginas do protótipo uma vez que é onde os estudantes vão extrair conhecimentos acerca da programação, bem como testá-los em seguida. As lições e as buscas são equivalentes ao desafio proposto no modelo conceptual.

Sempre que o estudante entre nesta página surge uma modal composta por slides com a teoria relativa à categoria escolhida. Alguns desses slides, para além da teoria, contêm exemplos úteis para a compreensão e resolução dos exercícios propostos, como podemos observar na figura 4.8. O estudante apenas deverá começar o desafio após ter absorvido cognitivamente todas as informações fornecidas e portanto apenas quando o estudante chegar ao último slide é que surgirá um botão para que se possa fechar a modal da teoria. Desta forma o estudante terá a possibilidade de efetuar uma pesquisa mais aprofundada dos tópicos abordados na internet se assim o desejar, melhorando as suas habilidades de autoaprendizagem.

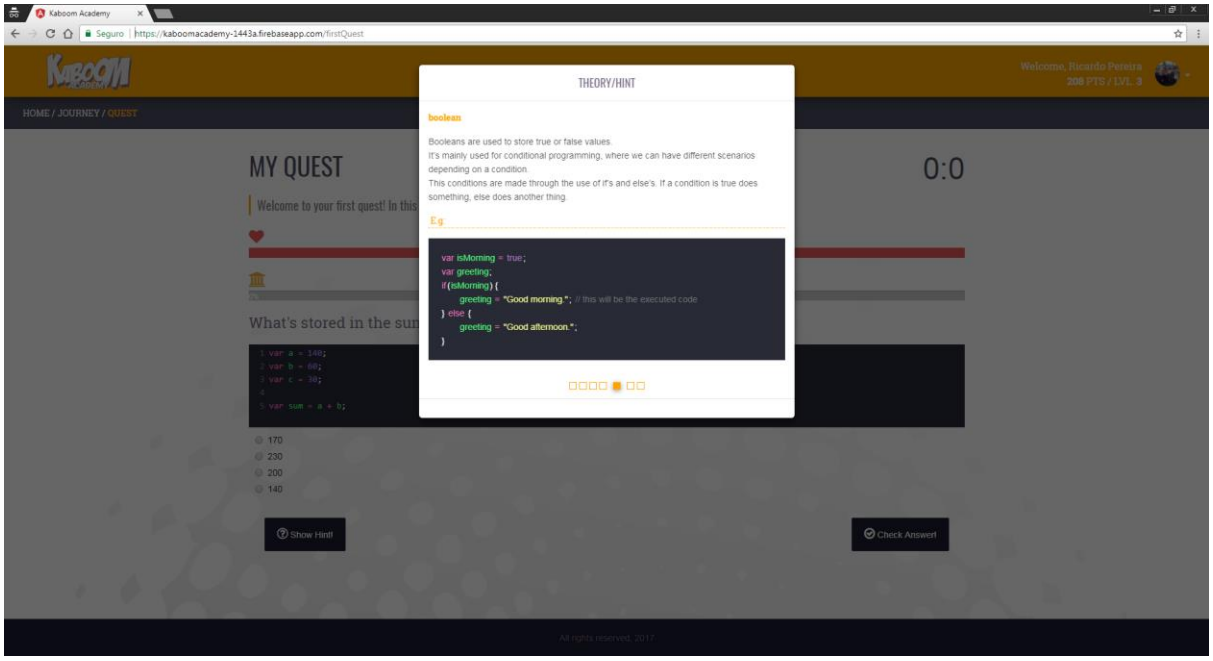


Figura 4.8: Página "Minha Busca" – Modal Teoria/Dica

Na figura 4.9 e 4.10, podemos observar que por cima da questão se encontra o estado atual do desafio, que é atualizado sempre que uma resposta é submetida. Na versão não gamificada o estudante irá confrontar-se com apenas números que indicam quantas questões foram respondidas, quantas foram acertadas e erradas. Na versão gamificada o estudante irá ter uma barra de vida e uma barra de experiência, bem como um temporizador que será inicializado após o fecho da modal da teoria, tal como foi proposto no modelo conceptual. Desta forma a experiência do estudante será jogável em vez de mandatária, uma vez que o mesmo irá ter a perceção que se encontra a realizar um desafio que se assemelha a um jogo motivando-o a não perder vida e a tentar preencher ao máximo a barra de experiência.

As três primeiras questões são do tipo "quiz" e são compostas por um texto, que na sua essência é a pergunta dirigida ao estudante, acompanhada de uma imagem com código pré-definido não editável que deverá ser analisada pelo estudante. Cada uma destas perguntas irá ter quatro possibilidades de resposta, onde apenas uma será a solução correta.

Caso o estudante tenha alguma dúvida acerca do exercício, terá disponível o botão "Mostrar Dica" que ao ser pressionado mostrará novamente a modal da teoria com os exemplos práticos que tem o intuito de fornecer dicas de como resolver o problema.

Sempre que o estudante desejar mudar a sua resposta, pode fazê-lo até que seja pressionado o botão “Verificar Resposta”. Quando este botão for pressionado, a resposta ao quiz será verificada e o estudante será remetido para a próxima questão.

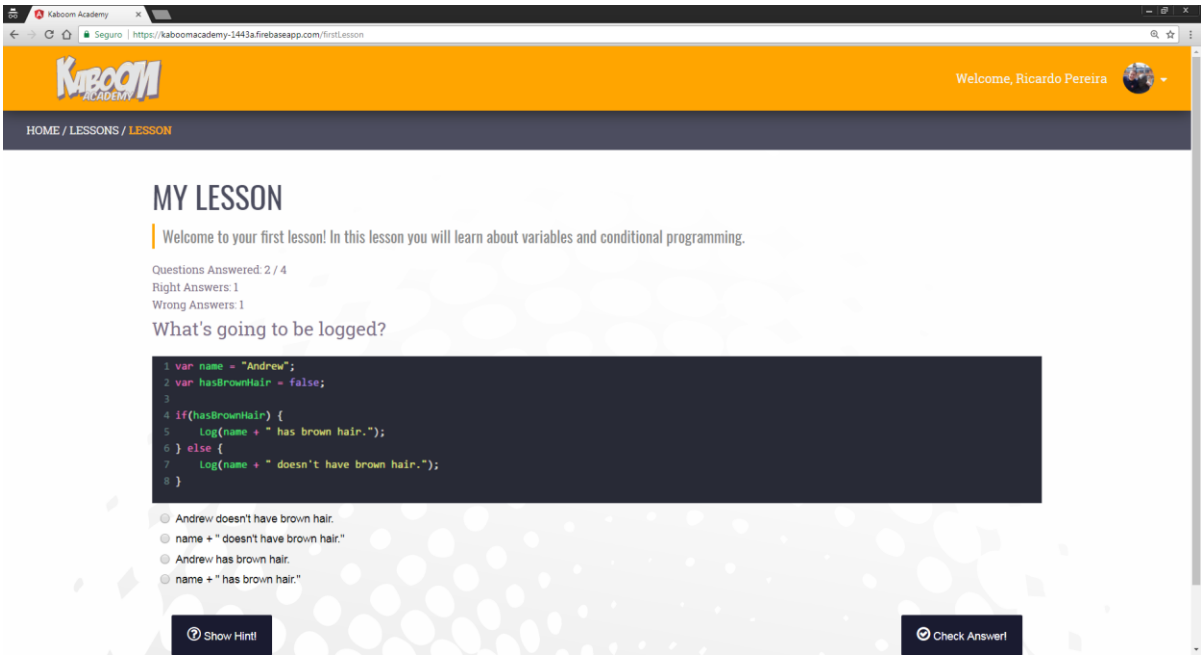


Figura 4.9: Página “Minha Lição” – Visualização do Progresso

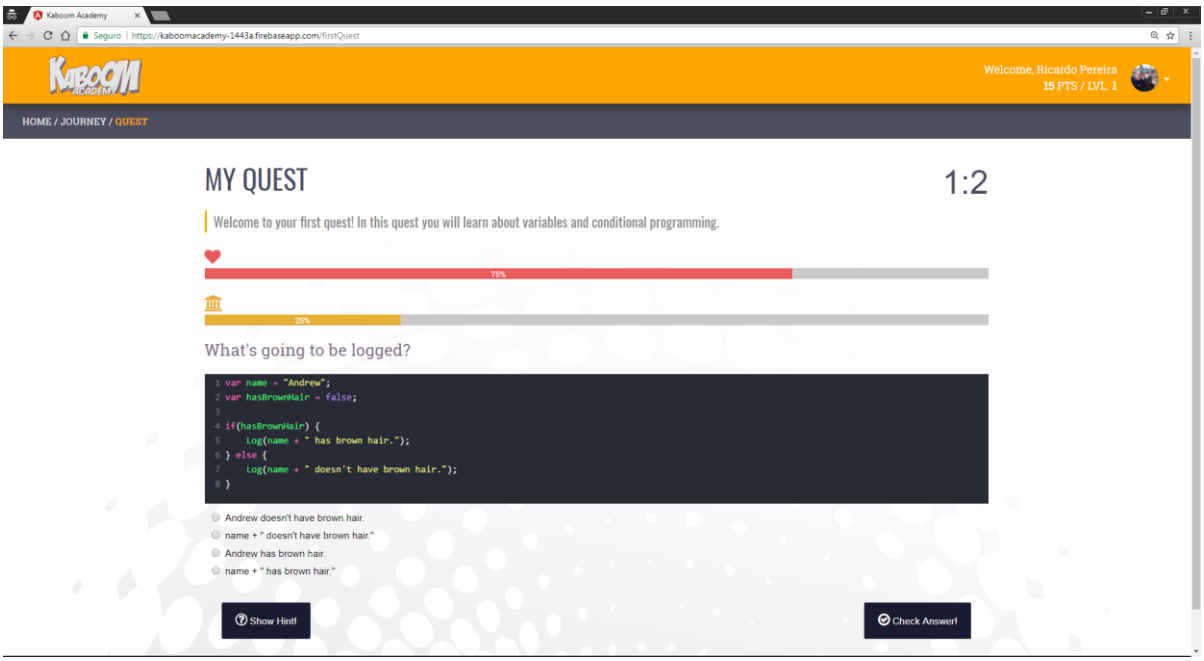


Figura 4.10: Página “Minha Busca” – Visualização do Progresso

Por fim, a última questão é do tipo editável, e é composta por uma pergunta em formato de texto, acompanhada por um bloco de código editável que será a resposta final do estudante a essa mesma questão.

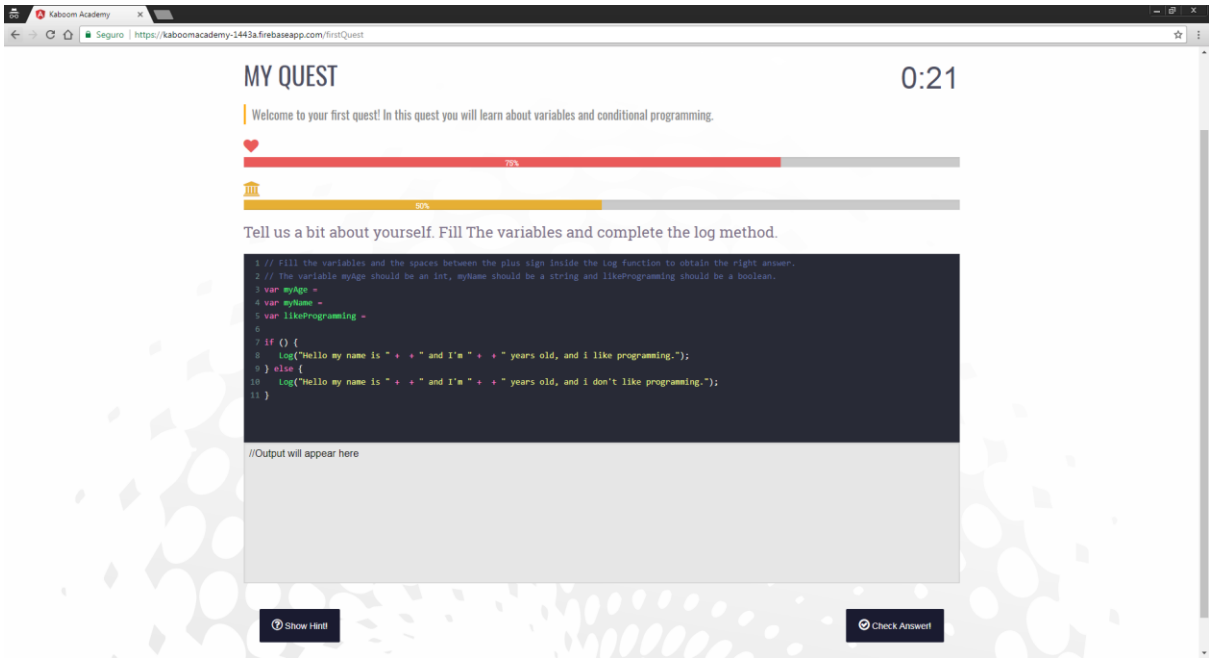


Figura 4.11: Página “Minha Busca” – Última Questão

Quando o estudante estiver satisfeito com o código previamente introduzido e tiver carregado no botão de verificação da resposta, o código será analisado e o output do mesmo passará para a área de texto abaixo da janela de código editável. No entanto, caso a resposta seja incorreta, aparecerá a indicação “Erro – Resposta incorreta” nessa mesma área de texto, tal como podemos observar na figura 4.11.

Para além da análise da resposta a esta última questão, após o clique no botão de verificação da resposta, irá surgir uma modal com o resumo do desafio e nota final obtida pelo estudante, como é exemplificado na figura 4.12 e 4.13. Esta modal terá o propósito de fornecer feedback instantâneo acerca dos resultados obtidos aquando a realização do desafio, bem como a divulgação dos crachás conquistados. Desta maneira o estudante irá sentir-se recompensado pelos seus bons resultados e irá sentir-se motivado e empenhado a continuar a enfrentar os restantes desafios propostos.

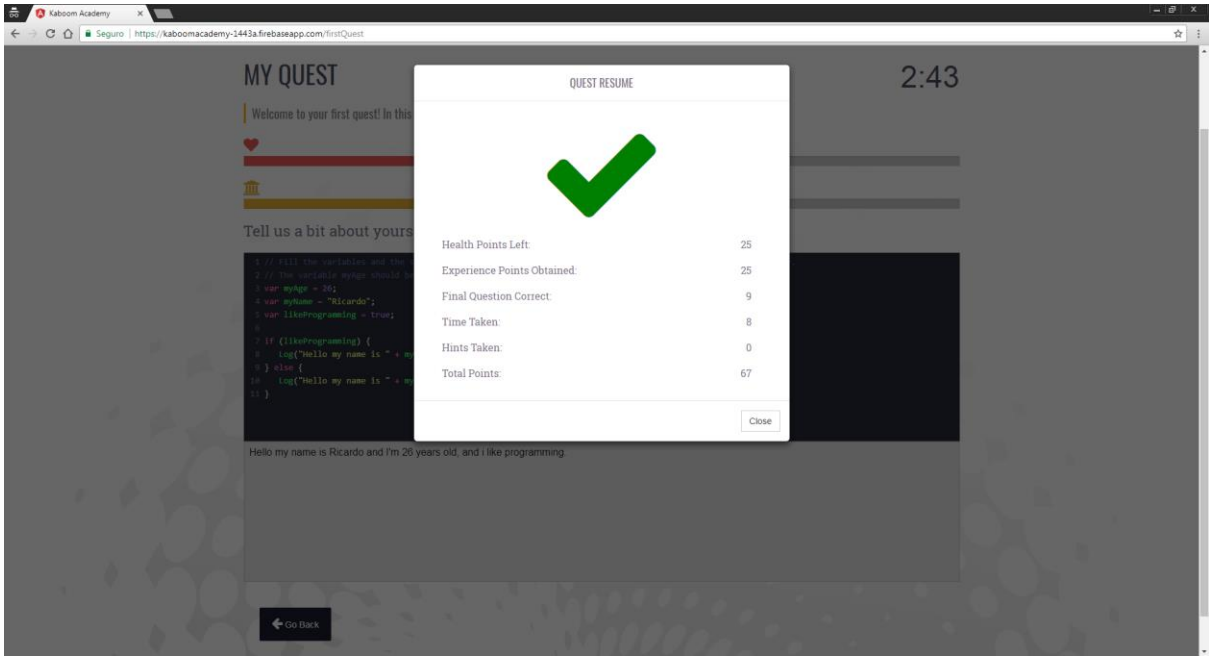


Figura 4.12: Página "Minha Busca" – Lição Aprovada

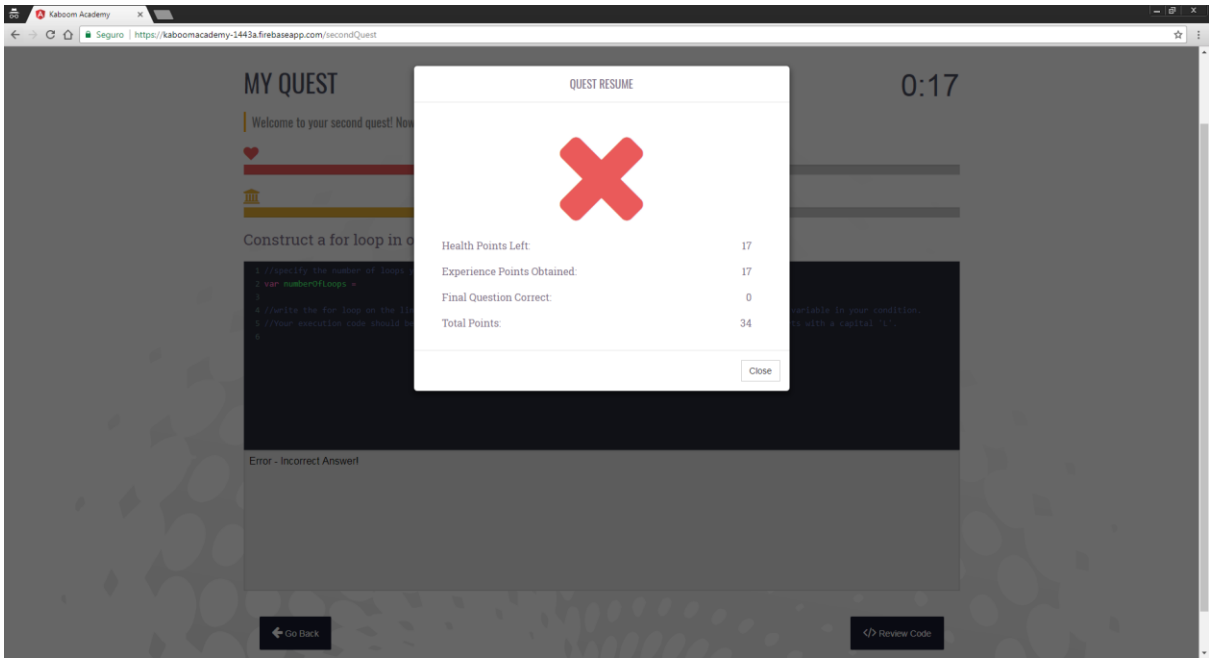


Figura 4.13: Página "Minha Busca" – Lição Reprovada

Na Figura anterior, podemos ainda verificar que após clique no botão "Fechar" da modal de resumo, o estudante passará a ter disponível dois botões, um para voltar para o ecrã anterior de seleção de desafios, e outro botão, "Rever Código", para reanalisar o bloco de código editável caso a resposta numa primeira instância tenha sido incorreta. Apesar do estudante poder reanalisar a última questão até acertar e output surgir corretamente na área de texto abaixo do bloco editável, esta funcionalidade serve apenas para propósitos de treino, uma vez que não

servirá para alterar a nota final obtida no desafio. Assim que o estudante tiver obtido a solução para a última pergunta, o botão de reavaliação do código desaparecerá e o estudante apenas poderá voltar para a página anterior.

4.2.2.4 Página "Minhas Estatísticas" e Página "Minhas Avaliações"

Tanto a página "Minhas Estatísticas", como a página "Minhas Avaliações", cumprem um dos requisitos funcionais, permitir ao estudante analisar todos os detalhes do seu desenvolvimento e progresso aquando o uso do protótipo.

A figura 4.14 descreve a página "Minhas Estatísticas", onde se verifica que o estudante em modo gamificado têm acesso às seguintes informações: (a) nível atual, (b) número de crachás desbloqueados, (c) número de pontos obtidos como consequência do desbloqueio dos crachás, (d) número de buscas começadas, (e) número de buscas completadas, (f) número de pontos obtidos pela conclusão de buscas, e por fim, o (g) número total de pontos que é a soma entre pontos provenientes de crachás e pontos provenientes de buscas.

Para além das informações referidas anteriormente, o estudante terá acesso a uma lista com todos os crachás disponíveis, sendo que os crachás desbloqueados aparecerão sem opacidade e com o seu nome por baixo, enquanto os crachás bloqueados surgirão opacos e com a descrição "Trancado". Caso o estudante carregue num crachá desbloqueado irá surgir uma modal com os seus detalhes, como podemos observar na figura 4.15.

Não existe nenhuma dica acerca do desbloqueio dos crachás uma vez que é desejável a promoção do espírito de aventura e exploração por parte do estudante.

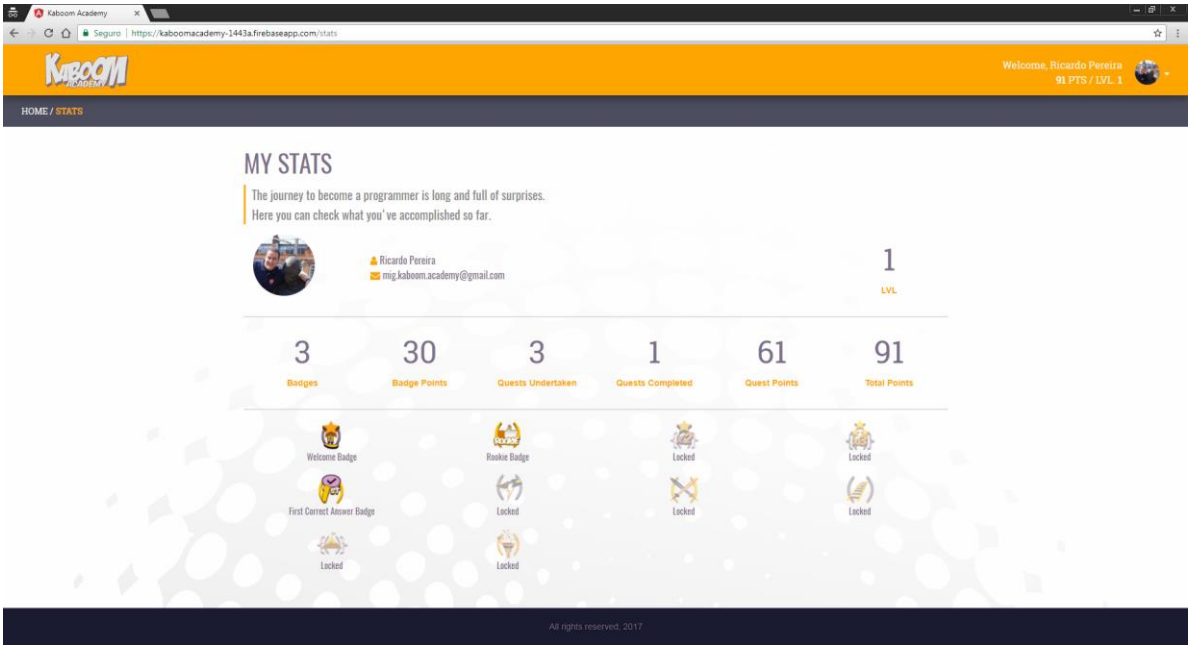


Figura 4.14: Página "Minhas Estatísticas"

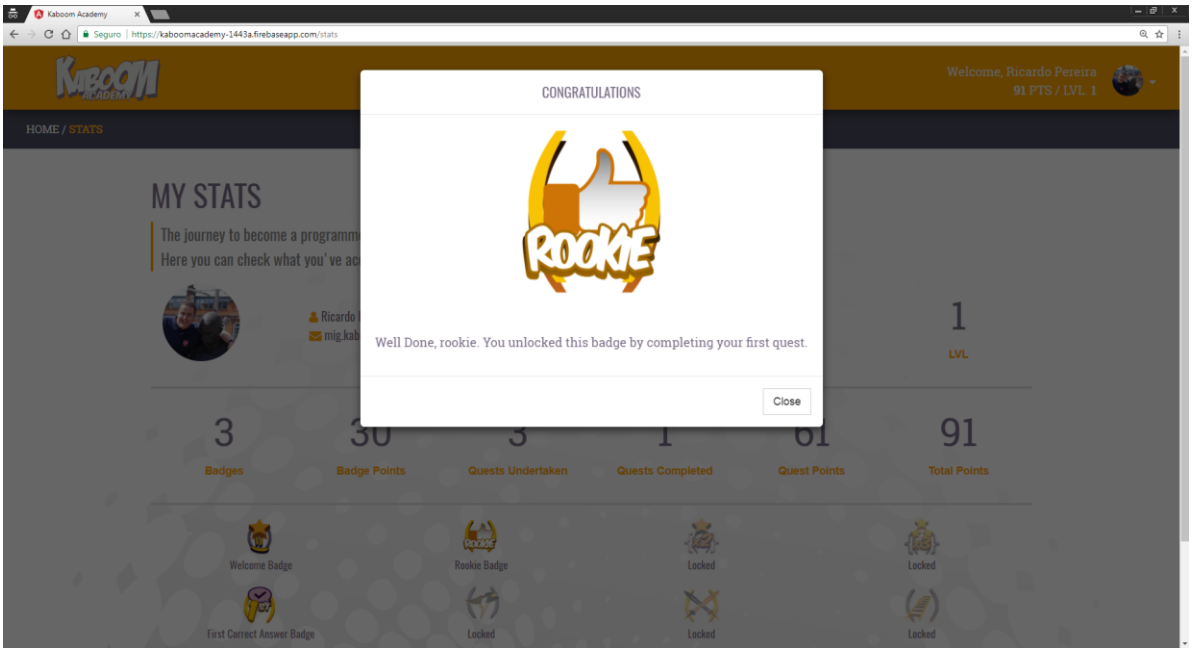


Figura 4.15: Página "Minhas Estatísticas" – Detalhes de um Crachá

Por outro lado, na figura 4.16, temos a página "Minhas Avaliações", onde podemos verificar que os estudantes em modo não gamificado tem acesso à informação relativa às melhores notas obtidas pela realização dos desafios referidos anteriormente. Para além dessa informação o estudante pode ainda verificar quantos testes tentou completar e quantos efetivamente completou, bem como a média geral dos três testes.

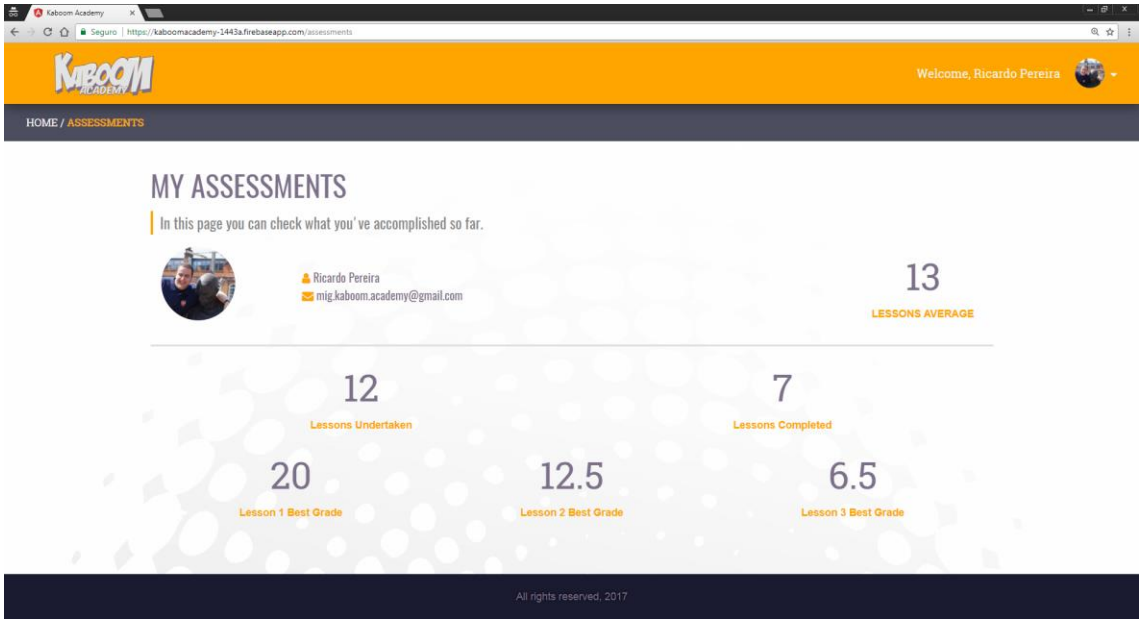


Figura 4.16: Página “Minhas Avaliações”

4.2.2.5 Página Tabela Classificativa

A página Tabela Classificativa, tal como foi dito anteriormente, apenas se encontra disponível para estudantes que estejam no modo gamificado, uma vez que serve para promover a competição e motivar a obtenção de melhores resultados. Nesta página os estudantes terão acesso a todas as estatísticas de outros estudantes que se encontrem no modo gamificado, tal como se pode observar na figura 4.17.

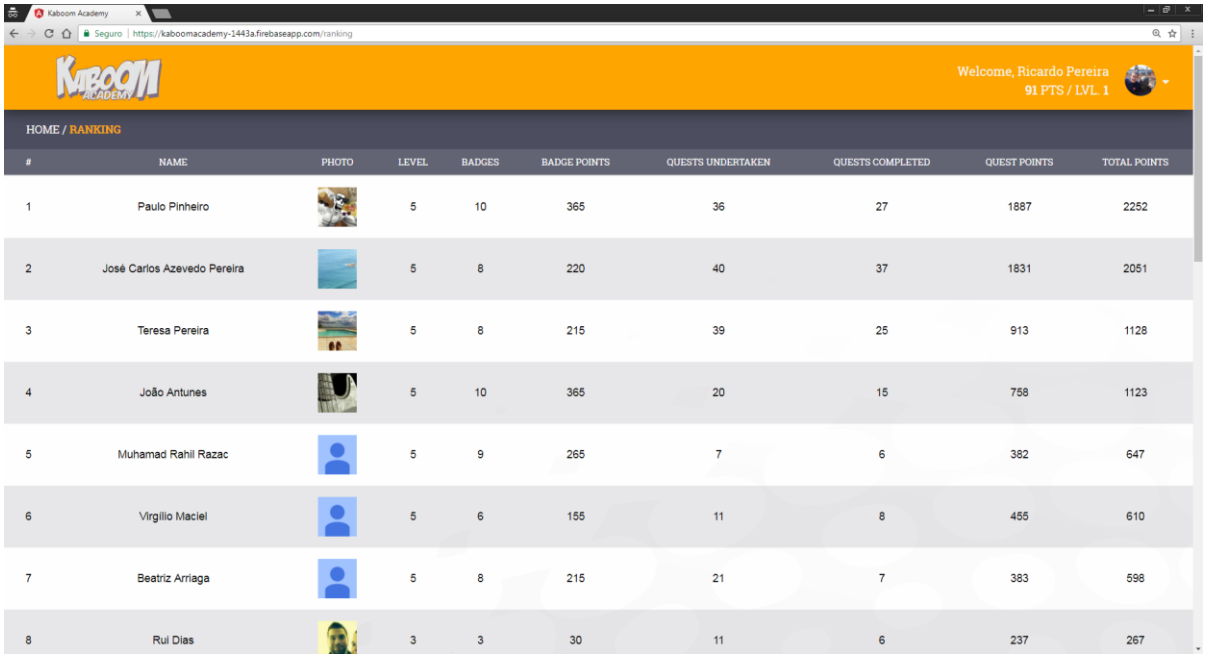


Figura 4.17: Página Tabela Classificativa

4.2.3 Outros Aspetos Relevantes

Como qualquer outra aplicação que contenha um mecanismo de autenticação, é necessário garantir que o utilizador possa fazer logout de modo a cancelar a sua autenticação e voltar à página de Login. O estudante terá a possibilidade de efetuar a ação de logout, em qualquer página e altura, através do widget existente no lado direito do cabeçalho do protótipo, que contém uma *dropdown* para esse efeito, como podemos verificar na figura seguinte.

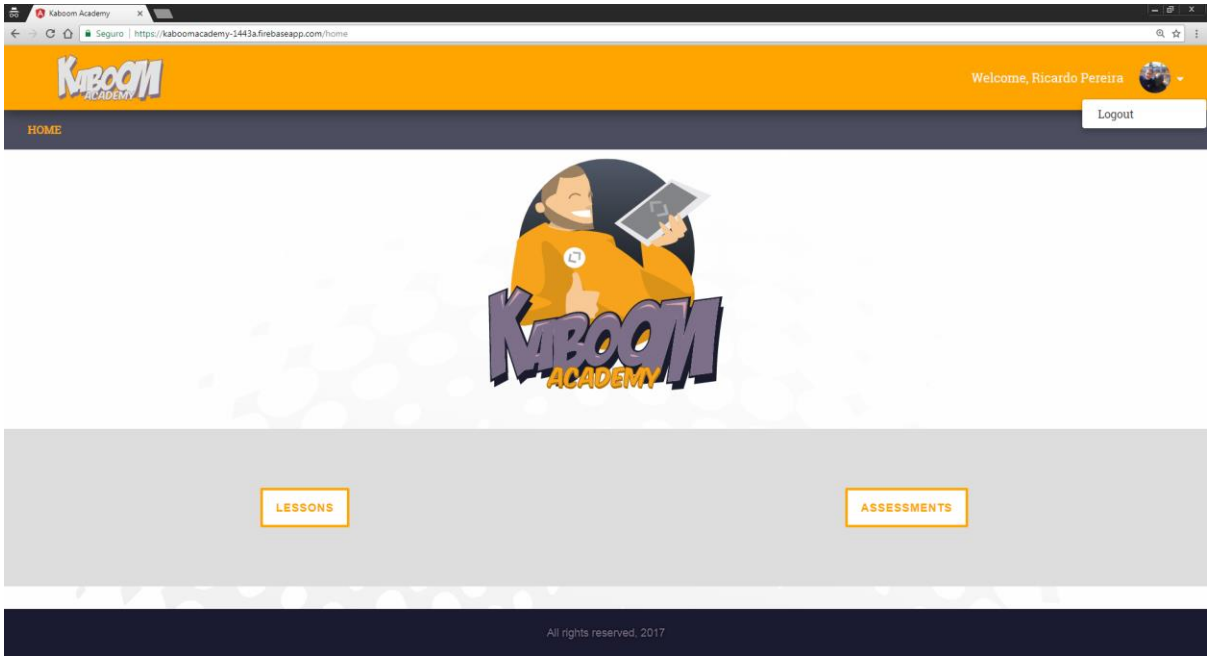


Figura 4.18: Ação de Logout

Um dos aspetos que também é importante garantir em qualquer aplicação é a navegação lógica e segura pelas suas diferentes páginas.

Para além das navegações entre as diversas páginas referidas nas subsecções anteriores, o estudante irá ter disponível em todas as páginas da área privada um *breadcrumb,* localizado no canto superior esquerdo abaixo do logotipo do protótipo, para que se consiga situar em relação à página inicial. Caso o estudante queria voltar a uma página anterior, ou no caso de estar na página de desafio e queira voltar para a página inicial, pode fazê-lo através de um clique numa das opções disponíveis no *breadcrumb,* como podemos observar na figura 4.10.

Por fim é necessário garantir que exista uma página que alerte o utilizador para o fato de estar fora do escopo de navegação lógica. Geralmente este cenário acontece quando é introduzido um *url* inválido. Existe, portanto, uma página em que o estudante se irá debater com a descrição "404: PAGE NOT FOUND" e um botão "Voltar atrás" que permitirá voltar para a página inicial e recomeçar novamente a navegação por um percurso lógico e válido.

Caso a navegação para essa página tenha sido proveniente da página login, ou seja, o estudante não estava autenticado, a ação de voltar remeterá o estudante de volta para a página de login.

No entanto, caso o estudante esteja autenticado, se navegar para uma página inválida, a ação de voltar atrás remeterá sempre o estudante para a página inicial, mas em alternativa, será possível realizar a ação logout, como é visível na figura 4.19.

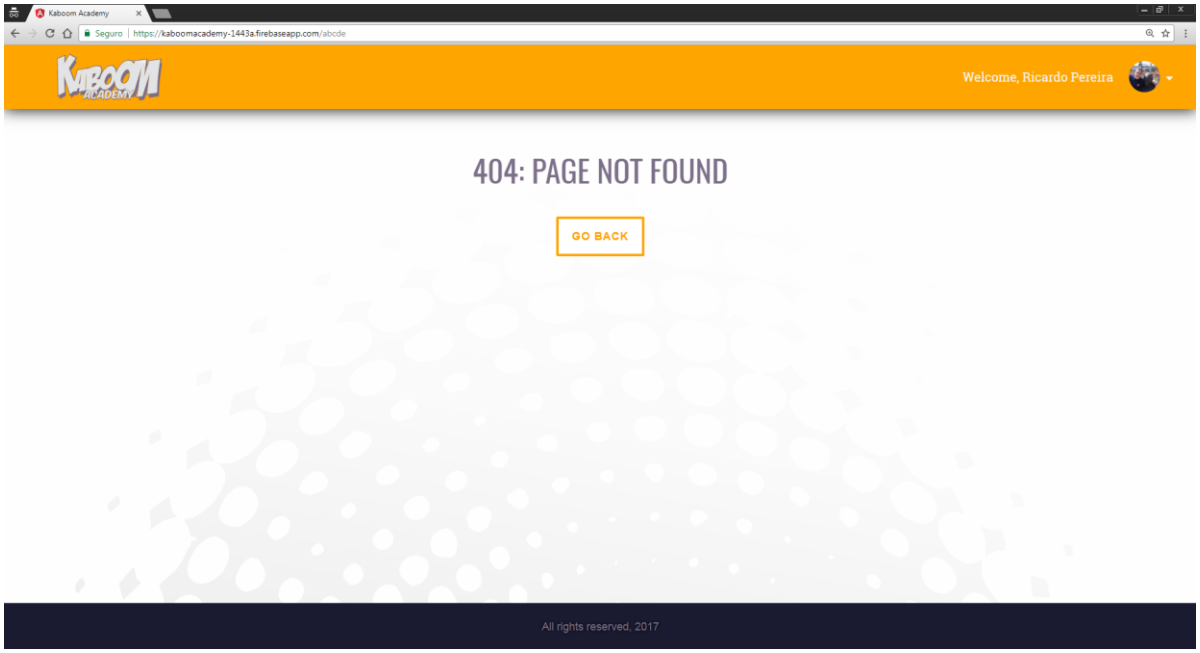


Figura 4.19: Página “404:PAGE NOT FOUND”

4.3 Arquitetura do Sistema

Este subcapítulo irá apresentar as escolhas tecnológicas envolvidas no desenvolvimento do protótipo Kaboom Academy, bem como os modelos de dados que foram definidos para tornar a base de dados o mais escalável possível, e por fim a estrutura aplicacional e todos os seus componentes.

4.3.1 Tecnologias Utilizadas

Em primeiro lugar, foi necessário definir que tecnologias seriam mais adequadas e robustas de implementar.

As duas principais tecnologias utilizadas na conceção do protótipo foram a *framework* Angular e a plataforma de desenvolvimento do Firebase. Para além destas, foram ainda utilizadas outras tecnologias, estruturadas na seguinte tabela:

Tabela 4.1: Tecnologias Utilizadas no Desenvolvimento do Protótipo

Tecnologia	Funcionalidade
Angular 4	*Framework* para desenvolvimento de aplicações *web*
Firebase	Plataforma de desenvolvimento de aplicações *web*
TypeScript	Linguagem de programação para o lado do cliente e servidor
JavaScript	Linguagem de programação para o lado do cliente
HTML	Linguagem de marcação para desenvolvimento de páginas *web*
CSS + BootStrap 3.3.7 + Font Awesome + Foundation	Auxiliares de design das páginas *web*
Jquery 1.12.4	Biblioteca para manipular HTML, CSS e Eventos em JavaScript
Node.JS	Interpretador de Código JavaScript no lado do servidor
NPM	Gestor de pacotes para desenvolvimento em JavaScript

Numa primeira instância foi necessário instalar o NPM e o Node.js. NPM é um gestor de pacotes para desenvolvimento em JavaScript, e consiste num cliente da linha de comandos onde passa a ser possível aceder a uma base de dados *online* de pacotes públicos de conteúdos em JavaScript. Por sua vez o Node.js é um ambiente de execução construído em JavaScript.

4.3.1.1 Angular

Angular é uma *framework* de código aberto orientada ao desenvolvimento de aplicações *web*. Esta escolha permite uma rápida e eficaz implementação de todos os ecrãs e da lógica *back-end* do protótipo.

A principal linguagem de programação utilizada no desenvolvimento sob a *framework* Angular é o TypeScript, que por sua vez é um superconjunto do JavaScript, e por esse motivo, esta linguagem adquire todas as funcionalidades das bibliotecas do JavaScript e ainda suporta outras funcionalidades adicionais. Entre as potencialidades do TypeScript, destaca-se o facto de ser possível ter uma estrutura de dados orientada a objetos, bem como o fato de ser possível escrever código tanto do lado do cliente como do lado do servidor, através do ambiente de execução Node.JS, o que não acontece com puro JavaScript.

Uma das primeiras noções que se deve ter acerca desta *framework* é que a mesma em conjunto com o TypeScript, usa sintaxe específica denominada por objetos *metadata* que fornecem configurações que apenas são interpretadas pelo Angular. Exemplos desses objetos são os *decorators @NgModule*, *@Component* e *@Injectable*, que serão detalhados em seguida.

A arquitetura Angular baseia-se na premissa de existência de módulos que ajudam na conceção de aplicações através de blocos de funcionalidade, reduzindo a abstração entre as ligações das diversas partes constituintes da aplicação.

Os módulos são representados pelo *decorator @NgModule* de modo a que o Angular consiga compilar e correr o código que o integra. Os módulos Angular servem essencialmente o propósito de agrupar os diferentes módulos da aplicação, bem como os diferentes componentes e serviços que o constituem. O *decorator @NgModule* recebe como possíveis configurações a (a) *tag* imports que é responsável pela importação de módulos, a (b) *tag* providers responsável pela declaração dos serviços disponibilizados pelo módulo, a (c) *tag* declarations responsável por declarar os componentes que integram o módulo, a (d) *tag* exports que tem o objetivo de declarar os componentes que poderão ser importados por outros módulos, e por fim a (e) *tag* bootstrap que apenas deverá estar disponível no módulo de raiz da aplicação e que tem o objetivo de declarar qual será o componente principal da aplicação para que o template associado a esse componente seja utilizado durante todo o ciclo de vida da aplicação.

Os componentes angular são representados pelo *decorator @Component* e tem como objetivo suportar e gerir os dados que são fornecidos aos templates da aplicação, que podem ser vistos como os ecrãs que serão disponibilizados ao utilizador através do *browser*. Cada componente é composto por um template *core*, e um ficheiro CSS para a sua customização, bem como uma classe responsável por agrupar os dados que serão fornecidos a esse mesmo template.

Assim como o *decorator* dos módulos, o *decorator @Component* ajuda o Angular a compilar o código relativo ao componente através das configurações presentes no objeto de *metadata* que tem como principais *tags*, a (a) *tag* selector pode ser vista como um exportador do template *core* do componente de modo a que o mesmo possa ser utilizado em templates de outros componentes, como por exemplo, o selector app-modal que permite a reutilização do componente modal em diversas páginas do protótipo, a (b) *tag* templateUrl que indica o endereço do ficheiro HTML a ser incorporado no componente, e por fim a (c) *tag* styleUrls que é um *array* com os endereços dos ficheiros CSS que serão utilizados pelo componente. Para além das configurações mencionadas, o *decorator @Component* identifica a classe imediatamente abaixo declarada como a classe do componente.

A classe de um componente é constituída pelas suas propriedades, um construtor e pelos métodos responsáveis pela lógica aplicacional onde são feitas alterações aos dados que são fornecidos aos templates.

Todos os componentes angular tem um ciclo de vida composto por *hooks* geridos pelo próprio Angular que fica responsável por criar, alterar ou remover os elementos do componente no DOM, que é essencialmente uma interface para a leitura de documentos HTML válidos, ou seja, quando uma página *web* é carregada, os *browsers* criam um modelo de objeto de documentos, para que sejam possibilitadas as interações por parte do utilizador com o mesmo para que a página possa ser manipulada. Dos *hooks* do ciclo de vida existentes destacam-se o ngOnInit e o ngOnDestroy, onde ocorre a inicialização e destruição do componente, respetivamente. Esses *hooks* podem ser capturados pelo próprio componente, através da explícita implementação do *hook* no momento da declaração da classe do componente, para que seja possível realizar ações extra definidas pelo desenvolvedor, como por exemplo a verificação de que o utilizador se encontra autenticado no momento em que o componente está a ser inicializada.

Através dos parâmetros do construtor de um componente é possível injetar serviços de modo a que haja uma manipulação mais dinâmica e estrutural das propriedades do componente. Como exemplo, temos um serviço de autenticação responsável por verificar se o utilizador está autenticado, que está presente em diversos componentes mas apenas se encontra declarado numa única classe alheia ao próprio componente.

Os templates presentes nos componentes servem de guião para a apresentação dos elementos que serão visíveis ao utilizador, e para além de serem escritos utilizando HTML e CSS, têm presente código específico do Angular denominado diretivas. Geralmente as diretivas surgem entre *tags* HTML e têm como objetivo a manipulação de elementos no DOM. Existem dois tipos de diretivas, as diretivas estruturais e as diretivas de atribuição. As primeiras têm o objetivo de alterar, adicionar ou remover elementos no DOM, alterando toda a estrutura do ecrã que é disponibilizado ao utilizador final. Como exemplo, temos a diretiva *ngFor que diz ao Angular para criar um certo número de elementos HTML de acordo com as instruções fornecidas. Portanto, se tivermos uma *tag* HTML <Tr *ngFor=”Let user of leaderboardusers”><tr> irão ser criadas tantas *table rows* quanto o número de utilizadores que estiverem presentes na tabela classificativa. Por outro lado, temos as diretivas de atribuição que servem apenas para adicionar dados a um elemento já existente no DOM, como por exemplo a diretiva ngModel.

Para além das diretivas, o Angular também fornece um forte mecanismo para manipulação de dados nos templates denominado vinculação de dados. Com este mecanismo torna-se fácil o estabelecimento da ponte entre os elementos HTML presentes nos templates e os dados presentes na classe dos componentes. Existem quatro tipos de vinculação de dados (a) interpolação, (b) vinculação de propriedade, (c) vinculação de evento e (d) vinculação de duas vias. Todos estes tipos de vinculações de dados são definidos no código HTML do template *core* dos componentes.

A interpolação é a mais simples de utilizar e é representada por {{propriedade}}, onde o valor da propriedade que está presente na classe do componente será disponibilizado no ecrã do utilizador. Por exemplo, o pedaço de código <td>{{user.name}}</td> irá colocar dentro de uma *table data* o valor da propriedade nome do utilizador.

A vinculação de propriedade é representada por [propriedade]=“propriedade_proveniente_da_classe”, e é utilizada com o intuito de passar

valores de propriedades da classe do componente para a propriedade HTML que se encontre entre parênteses retos, como por exemplo <img [src]=”user.photoUrl”>, onde a propriedade photoUrl do utilizador será vinculada à propriedade src da *tag* HTML img.

A vinculação de evento é representada por (evento)=“acção” e serve para vincular um método da classe do componente a um evento do DOM. Como exemplo, temos o evento de clique num botão <button (click)=”goToHomePage()”>Ir para a página Inicial!</button> que irá executar o método goToHomePage() disponibilizado pelo componente.

Por último, a vinculação de duas vias é representada por [(propriedade)]=“propriedade_proveniente_da_classe” e pode ser vista como a combinação da vinculação de propriedade com a vinculação de evento, uma vez que o valor da propriedade presente no componente é passado para a propriedade do elemento HTML, e caso ocorra um evento o valor da propriedade presente no HTML é passado para a propriedade presente no componente esmagando o seu valor original definitivamente. Exemplo da potencialidade deste tipo de vinculação é a sua utilização num elemento do tipo input <input [(ngModel)]=”user.status”> onde o valor original da propriedade estado do utilizador irá aparecer no campo de input da página disponibilizada através da diretiva ngModel, e conforme o utilizador for alterando esse campo, os valores da propriedade que se encontra presente na classe do componente e o valor da propriedade associada ao elemento HTML input, vão sendo alterados em tempo-real.

O Angular permite ainda a implementação de serviços que podem ser utilizados em diversos componentes da aplicação. Os serviços são representados pelo *decorator @Injectable* e são essencialmente classes que têm o objetivo de adicionar lógica aplicacional como por exemplo um serviço de autenticação.

Estes serviços são maioritariamente consumidos pelos componentes angular da aplicação através de injeção de dependências. A *framework* angular disponibiliza um mecanismo chamado Injetor que pode ser visto como um contentor de serviços que foram criados e que estão a ser utilizados na aplicação. Caso um serviço ainda não tenha sido instanciado, o Injetor fica responsável por criá-lo e adicioná-lo ao seu contentor antes de devolvê-lo aos componentes que o solicitarem. Quando os serviços tiverem sido criados e adicionados ao contentor do Injetor, os mesmos são passados para o construtor do componente como parâmetro e passam a

ficar disponíveis durante o ciclo de vida do componente. Segundo as boas práticas do Angular, todos os serviços devem ser declarados na configuração *providers* dos módulos angular.

4.3.1.2 Firebase

A segunda tecnologia mais importante utilizada no desenvolvimento do protótipo foi a plataforma de desenvolvimento *web* Firebase. Esta plataforma oferece inúmeros serviços para um rápido e eficaz desenvolvimento de aplicações *web*, e no desenvolvimento do protótipo foram utilizados apenas os seguintes serviços: (a) serviço de autenticação, (b) serviço de base de dados em tempo-real e o (c) serviço de hospedagem *web*.

Em primeiro lugar foi criada uma nova conta Google que ficaria associada à plataforma de desenvolvimento do Firebase. Através desta associação, foi possível obter uma apiKey, um domínio e um endereço do repositório de base de dados para que o protótipo angular desenvolvido pudesse aceder aos serviços do Firebase, garantido assim a integridade necessária para que apenas os utilizadores autenticados no protótipo conseguissem aceder aos dados.

O serviço de autenticação oferecido pelo Firebase possibilita a autenticação dos utilizadores utilizando apenas código do lado do cliente, abstraindo toda a complexidade do lado do servidor, complexidade essa que é gerida pelos próprios servidores do Firebase. O utilizador insere as credenciais de uma conta Google que possua, que por sua vez são passadas para os servidores do Firebase, que disponibilizam serviços *back-end* para verificação destas credenciais, que posteriormente retornam uma resposta acerca do estado do login. Caso o login tenha sido efetuado com sucesso, o utilizador passa a ter acesso à base de dados em tempo-real do Firebase.

Apesar de ser possível autenticar com contas Google, facebook, GitHub, Twitter ou através de um *email* personalizado ou ainda entrar em modo anônimo, optamos por utilizar apenas o método de login através de uma conta Google, uma vez que a maior parte da população possui uma. Através do login com uma conta Google, foi possível extrairmos informações básicas acerca do utilizador tais como o nome, email e endereço da foto associada à sua conta, informações essa que serviriam para criar uma entidade própria dentro do nosso protótipo através da inserção das mesmas na base de dados do Firebase.

É importante referir que após o primeiro login do utilizador, é criada uma entrada na plataforma de desenvolvimento Firebase respetiva ao *token* de acesso do utilizador. Através do ecrã de autenticação disponibilizado pela plataforma Firebase é possível desativar ou excluir a conta do utilizador, para que o mesmo não possa aceder aos diversos serviços do Firebase.

Era necessário arranjar uma instância de base de dados que conseguisse agregar todos os dados dos utilizadores bem como os dados relacionados com as funcionalidades do protótipo, e portanto, a opção escolhida foi a base de dados em tempo-real do Firebase uma vez que permitia a criação de uma base de dados na *cloud* dos servidores Firebase que podia ser acedida por todos os utilizadores autenticados no protótipo. O serviço de base de dados do Firebase oferece a possibilidade de criação de uma base de dados NoSQL formatada em JSON que fica alojada na cloud do Firebase.

Os dados presentes na base de dados são compatíveis em todas as plataformas, ou seja, um utilizador pode estar no *browser* de um computador ou no telemóvel ou tablet, que irá sempre ter acesso à mesma base de dados. Estes dados estão sempre sincronizados, sendo que quando a base de dados sofre uma mudança, seja pela inserção, remoção ou alteração dos dados nela presentes, é enviada uma notificação a todos os utilizadores que estejam conectados à mesma, para que todos os utilizadores tenham acesso à mesma informação em tempo-real.

Esta base de dados disponibilizada pelo Firebase segue o modelo não relacional definido por nós JSON, onde apenas são permitidas operações que possam ser executadas de forma rápida e direta, tornando a sua estrutura mais dinâmica.

Este serviço oferece ainda a possibilidade de definição de regras também escritas em JSON, onde é possível definir por exemplo as regras de leitura e escrita sobre a base de dados. Neste protótipo foi aplicada apenas a regra de autenticação, que possibilita apenas a leitura e escrita na base de dados a utilizadores que estejam autenticados com sucesso.

Havia a necessidade de arranjar um serviço capaz de disponibilizar o protótipo de maneira segura na *web*, para que todos os utilizadores pudessem usufruir das suas funcionalidades e para que fosse possível a obtenção de dados para chegarmos a uma conclusão acerca da hipótese proposta nesta dissertação. Optámos portanto pelo serviço de hospedagem do Firebase. Este serviço possibilita a hospedagem segura de todos os conteúdos na *web* através de uma simples instrução na linha de comandos com a ajuda do Firebase Cli, que é um *plugin* de linha de comandos disponibilizado pelo Firebase.

Assim, através do serviço previamente mencionado, foi possível garantir que todos os conteúdos desenvolvidos utilizando a *framework* Angular em conjunto com os serviços de autenticação e de base de dados do Firebase, ficassem hospedados na *web* através de uma conexão segura SSL num subdomínio do domínio firebaseapp.com.

4.3.1.3 Plugins

Para que fosse possível implementar certas funcionalidades, foram ainda adicionados *plugins* que se encontram disponíveis no GitHub, que é uma plataforma de disponibilização de código aberto, e que foram instalados através da linha de comandos com o apoio do gestor de pacotes NPM. Os *plugins* adicionados e as suas características são resumidas na seguinte tabela:

Tabela 4.2: *Plugins* Utilizados no Desenvolvimento do Protótipo

***Plugins* GitHub**	**Funcionalidade**
Angular Cli	Cliente de linha de comandos para criar aplicações Angular de raiz bem como as suas componentes, testar a aplicação através de um servidor local e compilar a aplicação.
Firebase Cli	Cliente de linha de comandos do Firebase para fazer *deploy* da aplicação para o ambiente de produção.
angularfire2	Biblioteca oficial para utilização de Angular em conjunto com Firebase nas aplicações *web*.
codemirror	Editor de texto implementado em JavaScript com o intuito de ser utilizado através do *browser*.
ng-block-ui	Prevenir que o utilizador interaja com a página *web* enquanto estiver a ocorrer um pedido assíncrono ao servidor.
ng2-codemirror	Componente Angular para utilizar o codemirror no HTML.

rxjs	Biblioteca para gerir eventos assíncronos através de observáveis.

Posteriormente serão fornecidos exemplos das aplicações dos *plugins* referenciados na tabela 4.2 e especificadas as suas funcionalidades de acordo com o fluxo aplicacional do protótipo.

4.3.2 Arquitetura da Base de Dados

Como foi dito anteriormente, optamos por uma base de dados não relacional disponibilizada através dos serviços do Firebase, como tal, passou a ser de extrema importância a definição de uma estrutura sólida e escalável dos elementos que a constituíam.

4.3.2.1 Modelo de dados

Foram definidos quatro modelos principais, que podem ser vistos como objetos ou entidades, tendo em conta o paradigma de programação orientada a objetos.

As três primeiras entidades são estáticas, uma vez que são imutáveis ao longo do tempo, isto é, não sofrem alterações através do código disponibilizado em back-end. Essas entidades são as seguintes: (a) Questão, (b) Resposta e (c) Crachá. A entidade Utilizador é a mais importante das entidades da base de dados, uma vez que é alvo de diversas modificações e por conter todas as informações essenciais dos estudantes e dos seus progressos.

Na figura 4.20, podemos analisar o atual modelo de dados de uma entidade do tipo Questão.

```
export class Question {
    content: string;
    imageUrl: string;
    rightAnswer: string;
}
```

Figura 4.20: Modelo Questão

Uma Questão é constituída por um atributo chamado "conteúdo", que contém a pergunta em formato de texto que será dirigida ao utilizador. Depois existe ainda o atributo "*url* da imagem", que é a localização da imagem que vai ser alvo de análise por parte do utilizador. As imagens das questões foram guardadas estaticamente na solução por uma questão de otimização, mas

poderiam estar alojadas na internet através de, por exemplo, o serviço de armazenamento do Firebase. Por fim, existe ainda um atributo chamado "resposta correta" que contém o texto correspondente à resposta correta dessa mesma questão.

Em relação às questões finais de cada exercício, as mesmas têm apenas preenchido o atributo "conteúdo" uma vez que não existirá uma imagem para analisar, mas sim um bloco de código editável em substituição, e também porque não haverá respostas selecionáveis, uma vez que não se trata de uma pergunta do tipo "Quiz".

Na seguinte figura podemos observar o modelo de uma Resposta:

```
export class Answer {
    content: string;
}
```

Figura 4.21: Modelo Resposta

Uma Resposta é constituída apenas por um atributo chamado "conteúdo", que irá conter uma possível resposta a uma questão em formato de texto. Cada questão irá estar associada a quatro diferentes respostas de cada vez, sendo que irá haver um cruzamento entre o atributo "resposta correta" de uma Questão e as quatro possíveis respostas de modo a ser obtida a validação de uma resposta correta. Em relação às questões finais esta entidade não existirá uma vez que não existe uma resposta correta padrão, mas sim o input introduzido pelo utilizador.

Em relação aos crachás, é possível verificar o seu modelo na figura 4.22.

```
export class Badge {
    description: string;
    points:number;
    url: string;
    unlocked: boolean;
    wasNotified: boolean;
    identifier: string;
    modificationDate: string;
}
```

Figura 4.22: Modelo Crachá

Como podemos verificar na figura acima descrita, um Crachá é constituído por diversos atributos cada um com a sua funcionalidade. O atributo "descrição" representa um texto explicativo acerca do motivo da conquista do crachá, enquanto que o atributo "identificador" fornece o nome do crachá apenas. O atributo "pontos" representa o número de pontos que serão

obtidos após a conquista do crachá. O atributo "*url*" indica o endereço da imagem do crachá para que o mesmo possa ser ligado ao HTML. Mais uma vez, as imagens dos crachás ficaram guardadas estaticamente na solução por questões de performance. Os atributos "desbloqueado" e "foi notificado" servem para indicar se o crachá foi conquistado pelo estudante, e se o estudante já foi notificado acerca da sua conquista, respetivamente. Por fim, temos o atributo "data de modificação", que é atualizado no momento da criação e conquista do crachá.

As entidades do tipo Crachá serão posteriormente clonadas para a entidade do utilizador que será explicada em seguida.

A entidade Utilizador pode ser descrita através da seguinte figura.

```
import { Badge } from "../../gamification/shared/models/badge.model";

export class User {
  //General Atributes
  name: string;
  mail: string;
  photoURL: string;
  gamificationModeSet: boolean;
  gamificationMode: boolean;
  secondQuestUnlocked: boolean;
  thirdQuestUnlocked: boolean;
  numberOfLogins: number;
  createDate: string;
  updateDate: string;
  //Gamified Attributes
  level: number;
  badges: number;
  badgePoints: number;
  badgeURLs: Array<Badge>;
  questsUndertaken: number;
  questsCompleted: number;
  questPoints: number;
  totalPoints: number;
  //Non-Gamified Attributes
  lessonOneGrade: number;
  lessonTwoGrade: number;
  lessonThreeGrade: number;
  lessonsAverage: number;
  lessonsUndertaken: number;
  lessonsCompleted: number;
}
```

Figura 4.23: Modelo Utilizador

Como se pode verificar através da figura anterior, a entidade do Utilizador engloba todos os atributos necessários para que seja possível usufruir de uma experiência gamificada ou não, aquando a utilização do protótipo.

Em primeiro lugar temos os atributos gerais que todos os estudantes deverão ter, como o nome, o *email*, a localização da sua imagem de Conta Google, a indicação do modo em que se encontra, gamificado ou não gamificado, o número de logins efetuados, a data de criação da entidade e a data de alterações, que será sempre atualizada quando houver alguma alteração a um dos atributos da entidade. A atribuição do modo gamificado a um estudante é efetuada tendo em conta o número de registos presentes na base de dados, sendo que para inscrições pares o estudante será alocado ao modo gamificado, caso contrário será alocado ao modo não gamificado.

Em seguida temos os atributos gamificados que serão sempre inicializados com o valor padrão zero, mas que apenas serão utilizados e modificados por estudantes que estejam a usufruir do modo gamificado. Dentro desta categoria temos o atributo nível que é o indicador do nível do estudante, o atributo crachás que indica quantos crachás o estudante conseguiu conquistar, o atributo pontuação de crachás que indica o número de pontos que foram obtidos através das conquistas dos crachás, o atributo endereços dos crachás que é um vetor que contém os possíveis crachás a obter por parte do estudante bem como as suas propriedades. Para além dos atributos mencionados temos os atributos relativos às buscas, onde existe um atributo que indica quantas buscas foram começadas e outro atributo que indica quantas buscas foram efetivamente completadas, e ainda um atributo que indica a pontuação obtida através das buscas. Existe ainda um atributo que indica a pontuação total do estudante que é a soma das pontuações obtidas através dos crachás com as pontuações obtidas através das buscas.

Por fim temos os atributos não gamificados que, tal como os atributos gamificados, são inicializados com o valor padrão zero e apenas são modificados por estudantes que estejam no modo não gamificado. Temos portanto, um atributo que indica a melhor nota obtida pelo estudante na primeira lição, um atributo que indica a melhor nota na segunda lição, um atributo que indica a melhor nota na terceira lição e ainda um atributo que indica qual a nota média das três lições. À imagem dos atributos gamificados temos ainda um atributo no modo não

gamificado que nos indica quantas lições foram iniciadas e outro atributo que nos indica quantas lições o estudante efetivamente completou.

4.3.2.2 Estrutura da Base de Dados

Uma vez que o serviço de base de dados disponibilizado pelo Firebase fornece a possibilidade de estruturar a base de dados ao abrigo de um modelo não relacional composto por nós, numa primeira instância definimos os nós principais que teriam as ramificações necessárias para compor todo o ecossistema do protótipo desenvolvido. Esses nós são (a) as questões, (b) as respostas, (c) os crachás e (d) os utilizadores. Podemos obter uma visão geral desta estrutura através da seguinte figura:

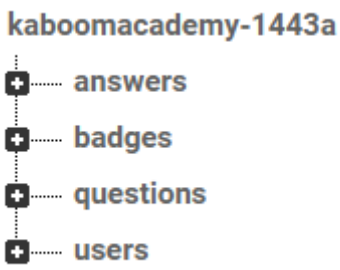


Figura 4.24: Principais Nós da Base de Dados

As questões e as respostas como mencionado anteriormente estão intimamente ligadas, e ambas são compostas pelas três lições/buscas existentes no protótipo, que por sua vez são compostas pelas quatro questões nelas presentes. Na figura 4.25 podemos observar estas composições tendo como exemplo a primeira lição/busca.

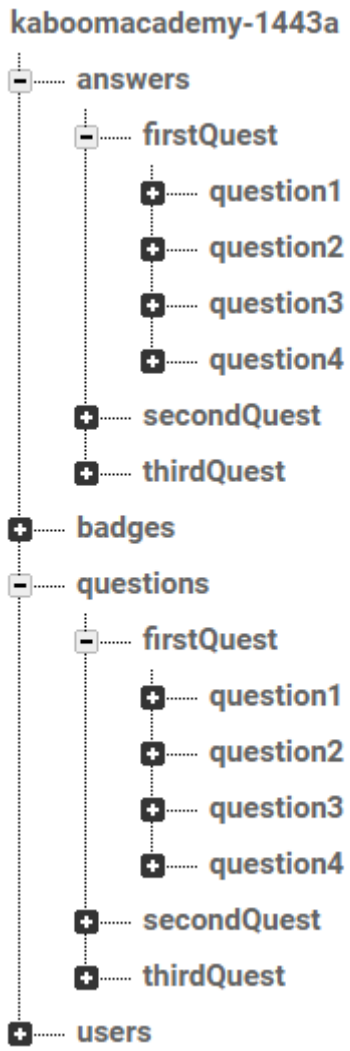


Figura 4.25: Estrutura das Questões e Respostas na Base de Dados

Dentro dos nós das lições/buscas podemos observar que temos as quatro questões, que representam os três quizzes e o desafio final. As questões que se encontram dentro do nó das questões são constituídas pelas entidades das questões referidas no subcapítulo anterior, e contém a pergunta a ser dirigida ao estudante, bem como o endereço da imagem a ela associada e a resposta correta, à exceção da última questão que apenas é composta pela pergunta uma vez que não existe imagem nem resposta correta a ser mapeada, tal como se pode observar na figura 4.26.

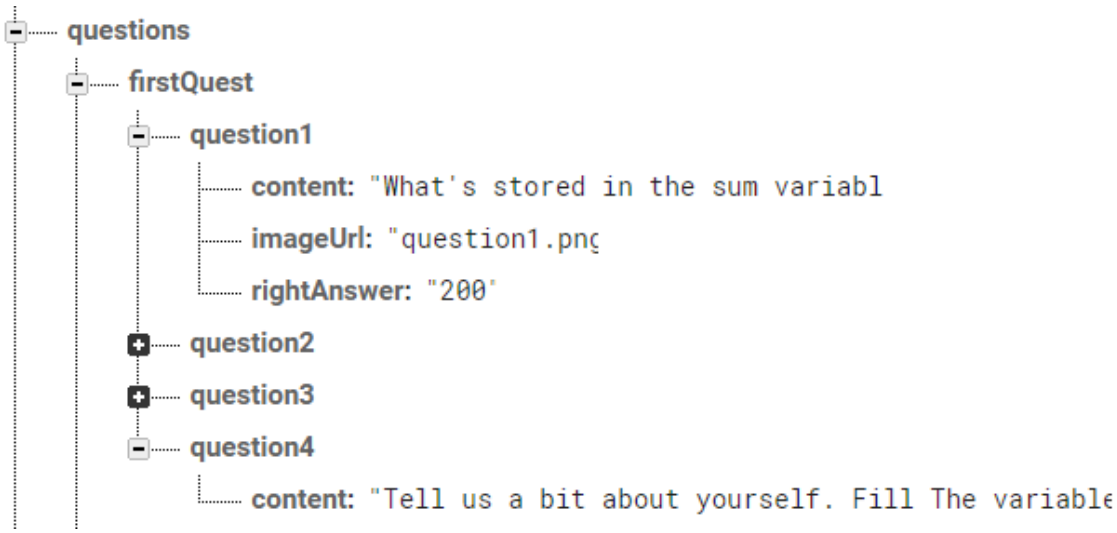


Figura 4.26: Estrutura do Nó "Questões"

Por sua vez, as questões presentes no nó das respostas estão diretamente relacionadas com as questões do nó das questões, e são compostas por quatro entidades do tipo resposta mencionadas no subcapítulo anterior, que serão utilizadas como opções de quiz no decorrer da lição/busca, à exceção da última questão que por não ser do tipo quiz, contém apenas o *template* que servirá de apoio à resolução do desafio final.

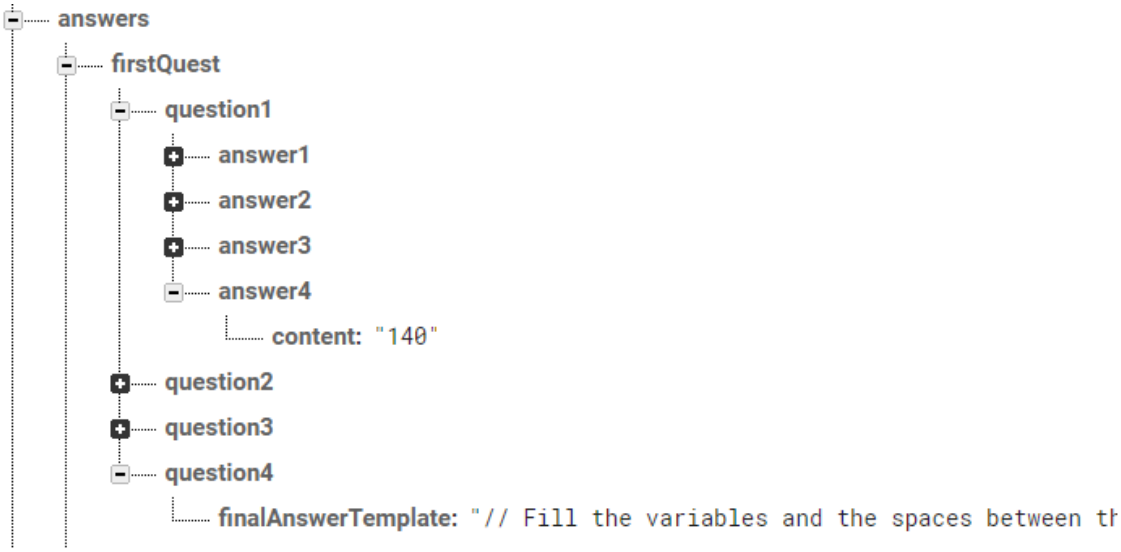


Figura 4.27: Estrutura do Nó "Respostas"

Em relação ao nó dos crachás, o mesmo é composto pelas entidades do tipo crachá referidas no subcapítulo anterior, onde são definidas todas as propriedades associadas aos mesmos. Estas entidades servirão de base para a construção do nó de cada um dos estudantes inscritos no

protótipo. Na figura 4.28 é possível visualizar a estrutura do nó dos crachás bem como o exemplo das propriedades associadas ao primeiro crachá.

Figura 4.28: Estrutura do Nó “Crachás”

Por fim, o nó dos utilizadores é composto pelas entidades do tipo utilizadores mencionadas no subcapítulo anterior, onde todas as propriedades referentes aos estudantes são guardadas. Dentro do nó de cada utilizador existirá um nó que será um clone do nó dos crachás, sendo que esse nó será modificado caso o utilizador se encontre na versão gamificada. O nome dos nós de cada utilizador são compostos pelos seus endereços de *email* menos os caracteres especiais como o caractere @ e ponto final, uma vez que não existem contas Google com o mesmo endereço, mas podiam haver utilizadores com o mesmo nome, esta foi a opção utilizada para que os utilizadores pudessem ser distinguidos.

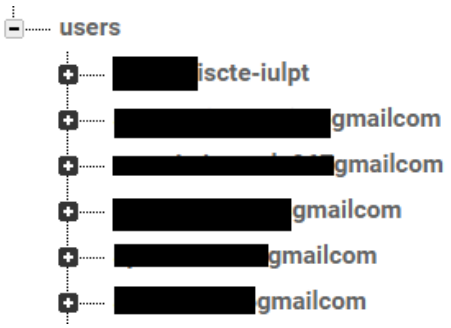


Figura 4.29: Estrutura do Nó “Utilizadores”

Figura 4.30: Exemplo de um Utilizador

Figura 4.31: Nó dos Crachás Associados a um Utilizador

4.3.3 Estrutura Aplicacional

No desenvolvimento do protótipo Kaboom Academy, foi utilizado como editor de código-fonte o Visual Studio Code e para que fosse possível a criação do protótipo era crucial estruturar todas as pastas e configurações necessárias. Como tal, usamos o *plugin* angular-cli, e através do comando "ng new nome-do-projeto" foi possível criar toda a estrutura necessária para um correto desenvolvimento de uma aplicação angular. Através deste *plugin* foi também possível a criação dos diversos componentes que constituíam os módulos do protótipo.

Na figura seguinte podemos obter uma visão de como a solução ficou estruturada na sua fase final.

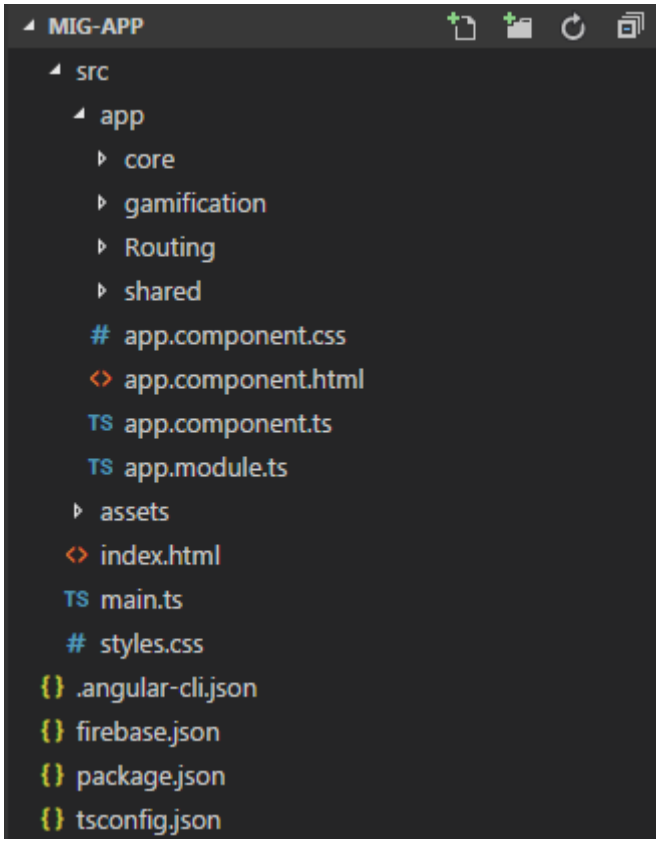


Figura 4.32: Estrutura do Código fonte

Podemos observar na figura anterior que existe uma pasta principal denominada "MIG-APP", que foi o nome inicialmente escolhido para o projeto, que por sua vez têm presente os diversos ficheiros de configuração do protótipo, bem como uma pasta "src" que contém todos os módulos, componentes e serviços do protótipo desenvolvido.

O ficheiro ".angular-cli.json" contém todas as configurações do protótipo angular, ficheiro esse que não foi alvo de alterações porque as configurações iniciais eram suficientes para o desenvolvimento do protótipo. O ficheiro tsconfig.json e o ficheiro firebase.json contém as configurações de typeScript e do firebase, respetivamente, e também não foram modificadas. No entanto, o ficheiro package.json contém todas dependências do protótipo ao nível dos *plugins* utilizados e a licença de desenvolvimento MIT do protótipo. Apesar deste último ficheiro ter sido modificado, essas alterações foram feitas de forma automática cada vez que os *plugins* eram instalados através da linha de comandos.

Existe um ficheiro styles.css que contém todos os estilos que são utilizados pelos diversos componentes do protótipo e uma pasta denominada assets onde se encontram todas as imagens que alimentam os diversos templates dos componentes.

O protótipo desenvolvido trata-se de uma aplicação de página única, onde existe apenas uma página durante todo o ciclo de vida aplicacional que vai sendo modificada através da gestão dos seus diversos templates que estão presentes nos componentes. Essa gestão é efetuada através da *tag* angular <router-outlet></router-outlet> presente no ficheiro app.component.html, que pode ser vista como um monitor que recebe os diversos templates dos componentes conforme especificações do código fonte.

Como podemos observar na figura 4.32, existe um componente que pode ser visto como o principal e primeiro a ser carregado no DOM e é denominado por componente app, que contém o header e o footer que estarão sempre presentes ao longo de todo o ciclo de vida aplicacional, bem como a *tag* <router-outlet></router-outlet> mencionada anteriormente. Este carregamento é efetuado através da *tag* <app-root></app-root>, que têm o propósito de carregar o template do componente principal do protótipo angular, presente no ficheiro index.html, que contém a declaração dos ficheiros JavaScript que serviram a aplicação com Jquery, bem como a declaração do uso de Bootstrap e de fontes personalizáveis que foram utilizadas ao longo da aplicação. O conteúdo deste ficheiro está presente na figura 4.33.

```
<!doctype html>
<html>

<head>
    <title>Kaboom Academy</title>
    <base href="/">
    <!--favicon-->
    <link href="favicon.ico" rel="icon" type="image/x-icon">
    <!--metas-->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv='cache-control' content='no-cache'>
    <meta http-equiv='expires' content='0'>
    <meta http-equiv='pragma' content='no-cache'>
    <!--Bootstrap and Jquery-->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHR
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCW
    <!--Custom css-->
    <link href="https://fonts.googleapis.com/css?family=Roboto+Slab:400,700" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Oswald:400,600" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/foundation/6.3.0/css/foundation-flex.min.css" rel="stylesheet" media="all" type="text/css">
</head>

<body>
    <app-root> </app-root>
</body>

</html>
```

Figura 4.33: Estrutura do ficheiro index.html

O ficheiro main.ts é um dos principais ficheiros uma vez que indica ao angular qual será o módulo principal do protótipo e portanto por onde a aplicação deverá começar a ser executada. No caso deste protótipo, o módulo principal será o módulo app que será detalhado em seguida. Na seguinte figura podemos analisar a composição do ficheiro main.ts.

```
import { enableProdMode } from '@angular/core';
import { environment } from './environments/environment';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

Figura 4.34: Estrutura do ficheiro main.ts

Como se pode analisar, este ficheiro serve para disponibilizar o protótipo em produção, ou seja, na *web*, para que esteja acessível a todos os utilizadores que introduzam o endereço do domínio do protótipo no *url* do *browser*. O método *platformBrowserDynamic* é fornecido pelo angular e serve para indicar que a aplicação deverá ser executada através de um *browser*, por sua vez o método *bootstrapModule* também fornecido pelo angular indica o módulo pelo qual a compilação da aplicação deverá começar. Este ficheiro é executado aquando a compilação do código para posterior *deploy* para produção.

Optamos pela estratégia de compilação Ahead-Of-Time em detrimento da compilação Just-In-Time. Numa estratégia de compilação Just-In-Time os componentes da aplicação são compilados no explorador do utilizador sempre que está a ocorrer o carregamento da aplicação, o que implica penalizações ao nível da performance como o prolongamento desnecessário do carregamento das páginas HTML, bem como impossibilita a deteção de erros nos templates HTML antes de os mesmos já terem sido carregados e o utilizador ter ficado com um erro escondido na consola de desenvolvedor fornecida pelos *browsers*. Por esse motivo, optamos por uma compilação Ahead-Of-Time onde todos os componentes da aplicação são compilados apenas durante o processo de construção da aplicação.

4.3.3.1 Módulos

Como podemos observar a partir da figura 4.32, o nosso protótipo é composto por cinco principais módulos: (a) o módulo App, (b) o módulo Core, (c) o módulo da Gamificação, (d) o módulo de Encaminhamentos e o (e) módulo Partilhado. Estes cinco módulo estão interligados através do módulo app e em conjunto definem na íntegra todas as funcionalidades do protótipo.

O módulo App, tal como foi referido anteriormente, é o módulo principal uma vez que é o ponto de ligação de todos os outros módulos e também dos componentes utilizados pelo protótipo. Estas ligações são feitas através da importação dos módulos e da declaração dos componentes que serão consumidos pela aplicação, como se pode verificar na figura 4.35.

```
//Base imports
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
//Modules
import { GamificationModule } from './gamification/gamification.module';
import { RoutingModule } from './Routing/routing.module';
import { SharedModule, sharedComponents } from './shared/shared.module';
import { CoreModule, CoreComponents } from './core/core.module';
//Main Component
import { AppComponent } from './app.component';

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    CoreModule,
    GamificationModule,
    SharedModule,
    RoutingModule
  ],
  declarations: [
    CoreComponents,
    sharedComponents,
    AppComponent
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 4.35: Estrutura do Módulo App

Em primeiro lugar foi importada a interface NgModule, importação essa que também se encontra presente nos restantes módulos do protótipo, de modo a que fosse possível a

declaração do *decorator @NgModule* e as suas consequentes configurações tais como as importações de módulos e declarações das componentes.

Neste módulo foram importados todos os módulos anteriormente referidos, bem como alguns módulos disponibilizados pela *framework* angular, tais como o módulo *browser* responsável por permitir que a aplicação corra num *browser web*, o módulo de formulários responsável por permitir o uso de formulários *web* para submissão de dados e o módulo HTTP responsável por permitir chamadas e respostas HTTP. Estas importações são a base de qualquer aplicação *web* desenvolvida em angular pelo que é fundamental as suas presenças.

Em relação aos componentes declarados, temos os componentes do módulo Core e os componentes do módulo partilhado, bem como o componente app. Desta forma o angular consegue reconhecer quais serão os componentes e os templates que serão utilizados ao longo do ciclo de vida aplicacional.

A *tag bootstrap* apenas está disponível neste módulo e serviu o propósito de declarar o componente app como sendo o principal uma vez que estará presentes em todos as páginas do protótipo.

Na figura 4.36 conseguimos obter uma visão geral do módulo *core* do protótipo.

```
import { NgModule } from '@angular/core';
//Modules
import { CodemirrorModule } from 'ng2-codemirror';
//Components
import { LoginComponent } from './publicArea/login/login.component';
import { HomeComponent } from './privateArea/home/home.component';
import { JourneyComponent } from './privateArea/journey/journey.component';
import { QuestComponent } from './privateArea/quest/quest.component';
import { StatsComponent } from './privateArea/stats/stats.component';
import { RankingComponent } from './privateArea/ranking/ranking.component';
//Shared Components
import { PagenotfoundComponent } from './shared/pagenotfound/pagenotfound.component';
//Services
import {LessonService } from './shared/services/lesson.service';

export const CoreComponents = [
  LoginComponent,
  HomeComponent,
  JourneyComponent,
  RankingComponent,
  StatsComponent,
  QuestComponent,
  PagenotfoundComponent
];

@NgModule({
  imports: [CodemirrorModule],
  exports: [CodemirrorModule],
  providers: [LessonService]
})

export class CoreModule { }
```

Figura 4.36: Estrutura do Módulo Core

Como é possível observar através da figura anterior, foi importado o módulo CodeMirror que é disponibilizado pelo *plugin* do codemirror, e tem o intuito de fornecer ao módulo *Core* a possibilidade de utilizar um editor de texto editável criado em javaScript no template HTML do componente relativo à página de desafio. Este módulo para além de importado para dentro do módulo *Core*, foi também exportado para que o mesmo ficasse disponível no módulo app, aquando a importação do módulo *Core*.

Em seguida foram adicionados a uma constante os componentes relativos ao (a) login, (b) página inicial, (c) jornada que equivale à página de seleção de desafios, (d) busca, que equivale à página de desafio, (e) página de estatísticas, (f) página tabela classificativa e (g) à página "404:PAGE NOT FOUND". Estes componentes são exportáveis uma vez que iriam ser declarados pelo módulo app e contém as informações relativas aos templates e lógica aplicacional das principais páginas do protótipo.

Em relação aos serviços disponibilizados pelo módulo, foi apenas declarado o serviço das lições é responsável pela obtenção das questões e respostas presentes na base de dados, bem como a alteração das propriedades do nó do utilizador que indicam o número de lições que foram iniciadas e o número de lições que foram completadas.

O módulo da gamificação é um dos módulos mais importantes do protótipo uma vez que integra toda a componente da lógica aplicacional gamificada apesar de na sua composição apenas disponibilizar o serviço de gamificação, como se observa na figura 4.37. Entraremos em detalhe acerca das funcionalidades deste serviço posteriormente.

```
import { NgModule } from '@angular/core';
//Services
import { GamificationService } from "./shared/services/gamification.service";

@NgModule({
  providers: [GamificationService]
})

export class GamificationModule { }
```

Figura 4.37: Estrutura do Módulo da Gamificação

O módulo de encaminhamentos representa todos os possíveis caminhos da aplicação, isto é, todos os endereços que o utilizador irá visualizar no campo de *url* do *browser* consoante a página em que se encontrar. As suas especificações podem ser visíveis através da seguinte figura.

```
import { NgModule } from '@angular/core';
//Base Modules
import { Routes, RouterModule } from '@angular/router';
//Components
import { CoreComponents } from "../core/core.module";
//Services
import { authGuardService } from "../shared/shared.module";

const routes: Routes = [
    { path: '', redirectTo: 'login', pathMatch: 'full' },
    { path: 'login', component: CoreComponents[0] },
    { path: 'home', component: CoreComponents[1], canActivate: [authGuardService] },
    { path: 'journey', component: CoreComponents[2], canActivate: [authGuardService] },
    { path: 'lessons', component: CoreComponents[2], canActivate: [authGuardService] },
    { path: 'ranking', component: CoreComponents[3], canActivate: [authGuardService] },
    { path: 'stats', component: CoreComponents[4], canActivate: [authGuardService] },
    { path: 'assessments', component: CoreComponents[4], canActivate: [authGuardService] },
    { path: 'firstQuest', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'first' } },
    { path: 'firstLesson', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'first' } },
    { path: 'secondQuest', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'second' } },
    { path: 'secondLesson', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'second' } },
    { path: 'thirdQuest', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'third' } },
    { path: 'thirdLesson', component: CoreComponents[5], canActivate: [authGuardService], data: { name: 'third' } },
    { path: '**', component: CoreComponents[6] },
];

@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})

export class RoutingModule { }
```

Figura 4.38: Estrutura do Módulo de Encaminhamentos

Este módulo pode ser visto como um *router* responsável por saber quais os pontos de entrada de cada uma das páginas, e como tal, importamos a componente das rotas e o módulo *router* do angular, bem como todos os componentes do módulo *core* que anteriormente exportamos, e o serviço de guarda de autenticação. É importante referir que o *router* foi exportado para que toda a aplicação conseguisse aceder ao mesmo, uma vez que o módulo de encaminhamento foi importado no módulo app.

As rotas são um tipo especial de configuração que recebe um único *array* de caminhos possíveis a seguir. Essas rotas por sua vez são passadas para o *router* angular através do método *forRoot*, que passará a ficar responsável por encaminhar o utilizador para os diversos componentes consoante as instruções fornecidas pelo código, ou pelo endereço introduzido no *url* do *browser*.

Podemos analisar através da figura 4.38 que os componentes associados a cada um dos caminhos possíveis é efetuado através do *array* das componentes que foi importado do módulo *core*, e que existem caminhos que têm os mesmos componentes associados, como por exemplo, o caminho da página das minhas estatísticas e o caminho da página minhas avaliações uma vez

que ambos partilham o mesmo componente por terem o mesmo *template* apesar das suas diferenças motivadas pelo modo gamificado e não gamificado.

Para além dos componentes é ainda importado e passado para cada um dos caminhos o serviço de guarda de autenticação, através do método disponibilizado pelo angular *canActivate*, que têm como único objetivo verificar que o utilizador se encontra autenticado quando entra numa página que estiver ao abrigo do serviço, e caso o utilizador não esteja devidamente autenticado, o mesmo será remetido para a página de login a fim de efetuar uma nova autenticação.

A *tag data* passada nas configurações dos caminhos relacionados com as buscas e com as lições servem para passar dados que são capturados na classe do componente. No nosso caso, passámos uma *string*, que é a representação de texto em código, correspondente à busca ou lição em que o utilizador se encontrava, uma vez que com este método era mais fácil identificar que perguntas e respostas seriam necessárias obter da base de dados, por exemplo, se for passada a *string* 'first' então o componente da busca/lição saberá que tem de obter as perguntas e respostas relativas ao primeiro desafio.

O caminho vazio '' e o caminho '**' são especiais, sendo que o primeiro indica que se o utilizador apenas colocar o endereço do domínio do protótipo no url do *browser* será encaminhado para a página de login, e o segundo indica que se for colocado um caminho não especificado, como por exemplo, "domínio\abcde" no *url* do *browser* o utilizador será encaminhado para a página "404: PAGE NOT FOUND".

```
<nav class="breadcrumb">
  <a class="breadcrumb-item" routerLink="/home">Home /</a>
  <span class="breadcrumb-item active">Ranking</span>
</nav>
```

Figura 4.39: Exemplo de Encaminhamento para a Página Inicial

Na figura anterior podemos observar um exemplo de encaminhamento para o ecrã principal através de uma técnica de navegação denominada *breadcrumb*, onde surgem migalhas com os possíveis encaminhamentos, sendo que a localização atual destaca-se sempre com uma cor

mais intensa. No exemplo fornecido, o utilizador encontrasse na página tabela classificativa, sendo que se carrega-se na migalha "HOME", seria encaminhado para o componente da página inicial, através da *tag* angular routerLink que obriga a aplicação a solicitar o endereço correspondente ao valor que lhe está associado ao *router*, destruindo o componente da página tabela classificativa ao mesmo tempo que injetaria o template da página inicial na *tag* <router-outlet></router-outlet>, fazendo-o surgir no *browser* do utilizador. Esta *tag* que se encontra presente no componente principal app, é responsável por disponibilizar todos os templates existentes nos componentes, através das indicações presentes no módulo de encaminhamentos.

Tanto o módulo da gamificação como o módulo de encaminhamentos não representam nenhum componente na sua estrutura, apesar de serem de extrema importância para o funcionamento de todo o protótipo.

Por fim, o módulo partilhado é constituído pelos elementos que são partilhados por todas as páginas do protótipo. Na figura 4.40, temos uma visão geral de como o módulo é composto.

```
import { NgModule } from '@angular/core';
//Modules
import { AngularFireModule } from 'angularfire2';
import { BlockUIModule } from "ng-block-ui";
//Configs
import { firebaseConfig } from "./services/firebase.config";
//Components
import { HeaderComponent } from './header/header.component';
import { FooterComponent } from './footer/footer.component';
import { ModalComponent } from './modal/modal.component';
//Services
import { AuthGuard } from './services/authguard.service';
import { AuthService } from './services/auth.service';
import { BlockUIService } from './services/blockui.service';

export const sharedComponents = [HeaderComponent, FooterComponent, ModalComponent];

export const authGuardService = AuthGuard;

@NgModule({
    imports: [
        AngularFireModule.initializeApp(firebaseConfig),
        BlockUIModule
    ],
    providers: [
        AuthService,
        AuthGuard,
        BlockUIService
    ],
  exports: [BlockUIModule],
})

export class SharedModule { }
```

Figura 4.40: Módulo Partilhado

Neste módulo foram importados os módulos do *plugin* BlockUi e do AngularFire, sendo que o módulo BlockUI também teve de ser exportado para que o módulo app o reconhecesse.

O módulo BlockUI permitiu a colocação de uma cortina transparente com a indicação *loading*, no *browser* do utilizador, enquanto estivessem a ocorrer pedidos assíncronos à base de dados, para que o utilizador não pudesse efetuar qualquer ação.

O módulo do AngularFire juntamente com o ficheiro de configuração, que se encontra contido no objeto firebaseconfig, que contém a apiKey, o domínio e o endereço do repositório de base de dados do Firebase permitiu que o protótipo tivesse acesso a todos os serviços disponibilizados pelo Firebase. A criação e inicialização da aplicação Firebase foi feita através do método disponibilizado pelo Firebase, *initializeApp*, que garante uma conexão segura aos servidores do mesmo. Uma vez que este módulo apenas serviu o propósito da criação da instância do Firebase, não foi necessária a sua exportação.

Foram importados e consequentemente exportados num *array*, os componentes referentes ao cabeçalho, rodapé e da *modal*, de modo a que os mesmos pudessem ser importados pelo módulo app, e portanto disponíveis ao longo de todo o ciclo de vida aplicacional.

Como serviços disponibilizados, temos o serviço de autenticação que foi de extrema importância para a gestão de autenticações dos utilizadores, o serviço de guarda de autenticação, que foi exportado para que pudesse ser importado no módulo de encaminhamentos como explicado anteriormente, e ainda o serviço do BlockUI que é responsável por habilitar e desabilitar a cortina transparente de *loading*.

4.3.3.2 Componentes e Serviços Gamificados

Neste subcapítulo iremos aprofundar os componentes que foram alvos de gamificação bem como o serviço que permitiu tornar a experiência do estudante gamificada.

Tendo em conta que o estudante estará a usufruir do modo gamificado, após o seu login na única página disponibilizada pela área pública, o mesmo será remetido para a sua primeira experiência gamificada, a página inicial. A página inicial gamificada, foi alvo de modificações de extrema relevância a nível visual, uma vez que os simples botões de acesso às diversas páginas do protótipo foram substituídas por ícones especificamente desenvolvidos para serem

o mais semelhantes a um videojogo. Estes ícones têm cores vibrantes e uma animação que proporcionará ao estudante a primeira sensação de que não se encontra numa simples aplicação, mas sim num jogo, tal como podemos verificar na figura 4.5.

Para além dos ícones gigantes o estudante irá sentir-se imerso no protótipo pelo fato de existir um cabeçalho permanente com a indicação do seu nome, nível e pontos correspondentes à sua participação no protótipo.

Para aumentar a motivação e incentivar o uso do protótipo, o estudante irá receber um crachá assim que entre pela primeira vez nesta página. Assim, o aumento da curiosidade e o sentimento de exploração irão emergir fazendo com que o estudante se sinta mais predisposto a utilizar o protótipo para aprender a programar.

Sempre que o estudante entrar nesta página irá ser verificado se o mesmo se encontra no topo da tabela classificativa a fim de receber o respetivo crachá. Esta verificação é efetuada através do serviço de gamificação que têm um método próprio para o efeito.

```
checkIfGotToTheTopOfLeaderboard(authState, userRef) {
    let leaderboardRef = this.getLeaderboardUsers();
    leaderboardRef
        .first()
        .subscribe(leaderboardUsers => {
            if (leaderboardUsers) {
                userRef
                    .first()
                    .subscribe(user => {
                        if (user) {
                            if (leaderboardUsers.length > 1 &&
                                    leaderboardUsers[0].mail === user.mail &&
                                        leaderboardUsers[0].totalPoints > leaderboardUsers[1].totalPoints) {
                                this.unlockBadge(authState, "9-on-top.png");
                            }
                        }
                    });
            }
        });
}
```

Figura 4.41: Verificação do Topo da Tabela Classificativa

Este método recebe os detalhes da autenticação do estudante que foram previamente obtidos pelo serviço de autenticação do Firebase e a referência do estudante que é equivalente ao seu nó na base de dados. Ambas as variáveis foram obtidas previamente através do serviço de autenticação desenvolvido para esse efeito. A obtenção do nó do estudante é efetuada através da instrução "af.database.object('/users/' + af.auth.auth.email), que utiliza o serviço AngularFire para obter o nó da base de dados através do caminho "/users/endereço de email

presente no token de autenticação do estudante fornecido pelos serviços de autenticação do Firebase". Em seguida obtemos todas as referências dos nós de todos os estudantes que se encontrem no modo gamificado e com mais do que zero pontos, ordenados descendentemente pela sua quantidade de pontos, e colocamo-los numa lista. Caso o nó do estudante atual tenha o mesmo *email* que o email do primeiro estudante da lista e mais pontos do que o segundo estudante da lista, será desbloqueado o crachá referente à conquista do primeiro lugar no topo da tabela.

Deste modo o estudante poderá ver o seu esforço recompensado, sabendo que se destacou dos restantes elementos do curso pela obtenção de melhores resultados. Todos os crachás conquistados surgirão numa modal como é demonstrado na figura 4.15.

O estudante terá o livre arbítrio de escolher qual a próxima página a frequentar e caso seja clicado o ícone da jornada, o mesmo irá ingressar numa página que terá um forte impacto visual uma vez que conterá três castelos sem qualquer descrição para que os mesmos sejam alvo de conquista e exploração.

Neste componente existe apenas um método que é responsável por verificar se o segundo e terceiro castelo já se encontram desbloqueados. Este método vai ao serviço de autenticação verificar se o estudante tem as propriedades "secondQuestUnlocked" e "thirdQuestUnlocked" e consoante os seus valores irá remover a opacidade do segundo e terceiro castelo, respetivamente, ao mesmo tempo que permitirá as suas conquistas, como pode ser visível na figura 4.7. Com esta metodologia, o estudante irá ter um acréscimo de curiosidade para saber o que poderá encontrar dentro de cada um dos castelos.

Dentro de cada castelo o estudante irá encontrar uma busca, ou desafio, que pode ser visto como um conjunto de tarefas a realizar para que o castelo seja conquistado. O componente relativo à página de desafio é um dos mais influenciados pela gamificação, uma vez que será onde o estudante passará a maior parte do seu tempo.

Assim que o estudante ingressar nesta página é chamado um método do serviço de gamificação que é responsável por incrementar o número de desafios começados, deste modo o estudante saberá quantas vezes tentou conquistar o castelo.

Em seguida é chamado o método do serviço de gamificação responsável por obter todas as questões da base de dados referentes ao castelo em que se encontra, bem como o método responsável por obter as respostas aos quizzes correspondentes. Desta forma apenas temos que

fazer a chamada assíncrona à base de dados uma vez, evitando atrasos ou constrangimentos relacionados com a conexão à internet. As questões e as respostas são guardadas em listas que por sua vez são utilizadas para alimentar o template do componente e fornecer ao estudante os dados relacionados com o conteúdo da questão, a imagem da mesma e os *bullet points* com as possíveis respostas.

A primeira ação que o estudante deverá fazer é ler a teoria associada ao desafio. Esta teoria, que se encontra presente numa modal como podemos observar na figura 4.8, pode ser vista como um guia para a conquista do castelo. Quando o estudante se sentir preparado para o desafio, o mesmo irá carregar no botão para o fecho da modal e ai terá o primeiro verdadeiro contato com os elementos mais fortes da gamificação, as barras de progresso e o temporizador.

As barras de progresso dão a possibilidade de fazer com que o estudante saiba instantaneamente em que patamar se encontra e como está a correr a sua experiência durante o desafio. Existem duas barras de progresso, uma que corresponde à vida do estudante que vai diminuindo caso seja respondida incorretamente uma questão, e uma barra de experiência que vai aumentando caso seja respondida acertadamente a uma questão. Este é um forte motivador uma vez que é desejável ter ambas as barras totalmente preenchidas de modo a obter o melhor resultado possível.

O temporizador por sua vez acrescenta o fator "stress" proporcionando um sentimento de pressão acrescida para que o estudante não se sinta demasiado confortável, pois saberá à partida que quanto melhor for o seu tempo, melhor será a sua pontuação. Este temporizador é iniciado logo após o clique no botão "fechar" da modal da teoria e começará do zero e não a partir de um valor fixo para não influenciar a resolução do desafio, uma vez que numa fase inicial de aprendizagem, é expectável alguma demora na resolução do mesmo. As barras de progresso e o temporizador podem ser observadas através da figura 4.10.

Entretanto irá ser começado o desafio através da resolução do primeiro quiz. O estudante irá ser confrontado com uma imagem e com quatro possíveis opções das quais deverá selecionar apenas uma. A qualquer momento poderá ser clicado o botão "Mostrar Dica" que fará surgir novamente a modal de teoria que contém exemplos que fornecem dicas de qual seria a resposta correta. Esta ação implica penalizações uma vez que cada vez que o estudante clique nesse botão, será incrementada uma variável que indica o número de cliques nesse botão, variável essa que será usada para penalizar o resultado final obtido.

Quando a resposta final tiver sido selecionada e o botão “Verificar Resposta” for clicado, é chamado um método da classe do componente do desafio responsável por verificar se a resposta foi acertada ou não, e atualizar as barras de progresso consoante esse fato. Para além da atualização das barras de progresso, é também atualizada a questão e as respostas presentes nos *bullet points* através dos elementos existentes nas listas dessas entidades inicialmente obtidas através da chamada à base de dados. Caso o estudante tenha chegado à quarta e última questão é atualizada a questão, sem uma imagem correspondente, mas por sua vez é utilizado o *plugin* codemirror para injetar um editor de texto no template da página com o texto existente no nó da base de dados correspondente à entidade da última resposta.

No momento em que o estudante acertar a sua primeira questão, é chamado um método do serviço de gamificação responsável por desbloquear o crachá referente a essa conquista e surgirá no ecrã uma modal com as características desse mesmo crachá como podemos verificar através da figura 4.15. Este crachá é de extrema importância uma vez que aumenta a motivação do estudante incentivando-o a prosseguir com o desafio inicial.

```
unlockBadge(authState: FirebaseAuthState, badgeUrl: string) {
    let userBadgesRef = this.getCurrentUserBadgeURLs(authState);
    userBadgesRef
        .first()
        .subscribe(badgesList => {
            if (badgesList) {
                let badgeWasUnlocked = false;
                let badgeToUnlock = null;
                badgesList.forEach(badge => {
                    if (badge.url === badgeUrl && badge.unlocked === false) {
                        badgeWasUnlocked = true;
                        badgeToUnlock = badge;
                    }
                });
                if (badgeWasUnlocked && badgeToUnlock) {
                    let badgeToUnlockRef = this.getBadgeObservableFromKey(authState, badgeToUnlock.$key);
                    badgeToUnlockRef.update({ unlocked: true, modificationDate: new Date().toUTCString()});
                    this.updateUserScore(authState, badgeToUnlock);
                }
            }
        });
}
```

Figura 4.42: Conquista de um Crachá

Este método gamificado referente à conquista de um crachá recebe como parâmetros os dados de autenticação do estudante bem como o endereço do nó do crachá que será desbloqueado, depois são extraídos da base de dados os crachás do estudante que já foram desbloqueados e colocados numa lista. Se o crachá ainda não tiver sido desbloqueado, iremos diretamente ao nó do crachá que se encontra na estrutura do estudante e alteramos as propriedades relativas ao

seu desbloqueio e à data em que a estrutura do nó foi modificada. Depois é chamado outro método do serviço de gamificação responsável por atualizar a pontuação do estudante, como se pode observar na seguinte figura.

```
updateUserScore(authState: FirebaseAuthState, badgeUnlocked?, questPoints?: number) {
    let userRef = this.getCurrentUser(authState);
    userRef
        .first()
        .subscribe(user => {
            if (user) {
                if (badgeUnlocked !== undefined) {
                    let updatedbadges = user.badges + 1;
                    let updatedbadgePoints = user.badgePoints + badgeUnlocked.points;
                    let updatedtotalPoints = user.totalPoints + badgeUnlocked.points;
                    userRef.update({ badges: updatedbadges,
                                     badgePoints: updatedbadgePoints,
                                     totalPoints: updatedtotalPoints,
                                     updateDate: new Date().toUTCString() });
                }
                if (questPoints !== undefined) {
                    let updatedquestsCompleted = user.questsCompleted + 1;
                    let updatedquestPoints = user.questPoints + questPoints;
                    let updatedtotalPoints = user.totalPoints + questPoints;
                    userRef.update({ questsCompleted: updatedquestsCompleted,
                                     questPoints: updatedquestPoints,
                                     totalPoints: updatedtotalPoints,
                                     updateDate: new Date().toUTCString() });
                }
                this.checkLevel(authState, userRef);
                if (user.badges > 1) {
                    this.checkIfGotToTheTopOfLeaderboard(authState, userRef);
                }
            }
        });
}
```

Figura 4.43: Atualização da Pontuação do Estudante

Este método uma vez mais recebe os dados de autenticação do estudante, e tem como outros possíveis *inputs* o crachá a ser desbloqueado e o número de pontos obtidos aquando a conclusão do desafio. Estes dois últimos parâmetros podem estar vazios, uma vez que este método é chamado após a conquista de um crachá ou após a conclusão de um desafio, uma vez que estas são as duas maneiras possíveis do estudante receber pontos e consequentemente subir de nível. Os dados de autenticação são utilizados para ir buscar o nó do estudante à base de dados e consequentemente efetuar a alteração das suas propriedades em relação ao número de crachás e pontos referentes a essa conquista, no caso de o *input* ter sido um crachá, ou alterar o número de desafios completados e o número de pontos referentes à conclusão do desafio, caso o *input*

recebido tenham sido esses pontos. Em ambos os casos é alterado o número total de pontos do estudante que será instantaneamente atualizado no cabeçalho do protótipo, bem como a data da modificação do nó do estudante. Em seguida é efetuada a verificação do nível em que o estudante se encontra através de outro método do serviço de gamificação, e ainda é verificado se o estudante alcançou o topo da tabela.

```
checkLevel(authState, userRef) {
    userRef
        .first()
        .subscribe(user => {
            if (user) {
                if (user.level < 5) {
                    let currentLevel = user.level;
                    let newLevel = this.checkNewLevel(user.totalPoints);
                    if (currentLevel < newLevel) {
                        userRef.update({ level: newLevel, updateDate: new Date().toUTCString() });
                        if (newLevel === 5) {
                            this.unlockBadge(authState, "8-ladder-of-success.png");
                        }
                    }
                }
            }
        });
}

checkNewLevel(userTotalPoints) {
    let userLevel: number;
    if (userTotalPoints >= 1 && userTotalPoints < 100) {
        userLevel = 1;
    } else if (userTotalPoints >= 100 && userTotalPoints < 200) {
        userLevel = 2;
    } else if (userTotalPoints >= 200 && userTotalPoints < 300) {
        userLevel = 3;
    } else if (userTotalPoints >= 300 && userTotalPoints < 400) {
        userLevel = 4;
    } else if (userTotalPoints >= 400) {
        userLevel = 5;
    }
    return userLevel;
}
```

Figura 4.44: Verificação do Nível do Estudante

O método gamificado para verificação do nível do estudante recebe como parâmetros os dados de autenticação do estudante e a referência do seu nó na base de dados. Em primeiro lugar é verificado se o estudante se encontra num nível inferior a cinco, que é o nível máximo, caso esse fato seja verdadeiro é então comparado o seu nível atual com o nível em que se irá encontrar, onde foi criado um algoritmo para que as pontuações de um a noventa e nove correspondessem ao primeiro nível, pontuações de cem a cento e noventa e nove correspondessem ao segundo nível e por ai adiante até à pontuação quatrocentos, valor esse que faria o estudante chegar ao quinto e último nível. Caso o nível atual fosse inferior ao seu

novo nível, iriamos alterar a estrutura do nó do estudante atualizando a propriedade nível com o novo nível e a data de modificação da entidade. Quando o estudante chegasse ao quinto e último nível, seria desbloqueado o crachá referente a essa conquista.

Quando o estudante tiver respondido às três questões do tipo quis, irá surgir o editor de texto mencionado anteriormente que têm o aspeto visível na figura 4.11. Este editor de texto deverá ser preenchido com código JavaScript, seguindo sempre a filosofia da matéria das três questões anteriormente apresentadas. Quando esta tarefa estiver terminada e o botão "Verificar Resposta" for clicado, é chamado um método na classe do componente que irá, em primeiro lugar, verificar se o *input* introduzido pelo estudante irá produzir o resultado esperado, isto é, se o motor de compilação de JavaScript iria conseguir compilar e executar o código. Caso a resposta tenha sido válida, irá surgir o *output* decorrente da execução do código numa área de texto adicionada para esse efeito, e será alterada uma propriedade na classe do componente de desafio para que sejam fornecidos os pontos referentes a esse feito recompensando o esforço do estudante. Se a resposta for incorreta irá surgir essa informação e o estudante não receberá qualquer recompensa. Em qualquer dos casos, a barra de progresso relativa à experiência iria aumentar ou a barra de progresso relativa à vida iria diminuir, e seriam fornecidos os devidos crachás referentes às conquistas dos castelos, bem como outros crachás surpresa. Na categoria de crachás surpresa temos o crachá referente à conquista do castelo com a barra de vida e de experiência totalmente preenchidas, e o crachá referente à conquista do castelo em menos de cinco minutos. É importante referir que os castelos poderão ser conquistados múltiplas vezes fornecendo liberdade ao estudante para usufruir ao máximo do protótipo promovendo o espírito de aventura e de competitividade, pois apenas assim seria possível conquistar todos os crachás e concluir em definitivo a experiência deste curso flash de ensino da programação.

Após a verificação de quais os crachás que foram desbloqueados é verificado quantos pontos o estudante recebeu pela conclusão do desafio. Esta verificação é efetuada através de um método existente na classe do componente do desafio, uma vez que envolve todas as propriedades nele presente. Em primeiro lugar foi adicionado um multiplicador de pontuação para que a mesma não tivesse valores fixos. Este multiplicador tem como valor zero vírgula trinta e três que é equivalente a um terço de qualquer tipo de ponto obtido. Em caso de conclusão com sucesso ou insucesso são sempre equacionados pontos ao nível da quantidade de vida que restou ao estudante, quantos pontos de experiência foram obtidos e a pontuação relativa à resposta correta do desafio final. A pontuação do desafio final varia consoante o

castelo que está a ser conquistado. No primeiro castelo a última questão vale vinte e cinco pontos, no segundo cinquenta e no terceiro cem. Caso o estudante conclua com sucesso o desafio, respondendo a pelo menos três questões corretamente, são ainda adicionados pontos relativos ao tempo despendido e serão retirados pontos consoante o número de dicas utilizadas. O temporizador começa com uma pontuação base de dez valores, que vai diminuindo consoante o tempo que for despendido, por sua vez, cada dica utilizada irá penalizar o estudante com menos cinco pontos em relação à pontuação final. A pontuação final da busca será o somatório destas pontuações, sendo que se o estudante falhar a conquista do castelo, não serão equacionadas as pontuações relativas ao temporizador e às dicas utilizadas. Como podemos observar através da figura 4.12 e 4.13, irá surgir uma modal com a descrição dos pontos obtidos no desafio. Desta forma, o estudante receberá feedback instantâneo acerca da sua *performance* o que poderá impulsionar a sua motivação para obter melhores resultados caso os mesmos não tenham sido positivos o suficiente na sua ótica, ou promover a sensação de satisfação por ter conseguido obter bons resultados, sendo que estes fatores são fundamentais em qualquer tipo de ensino.

Por fim é chamado um método do serviço de gamificação responsável por atualizar o número de desafios completados e a pontuação do estudante.

```
updateQuestsCompletedAndScore(authState, questPoints: number) {
    this.updateUserScore(authState, undefined, questPoints);
}
```

Figura 4.45: Atualização das Estatísticas do Estudante

Como podemos observar na figura anterior, este método recebe os dados de autenticação do estudante, bem como o número de pontos obtidos aquando a conclusão do desafio, dados esses que são passados para o método anteriormente explicado e presente na figura 4.43.

Quando o estudante fechar a modal referente aos resultados obtidos, irão ser fornecidas duas opções em detrimento das opções de “Mostrar dica” e de “Verificar Resposta”. Será possível regressar à página de seleção de desafios através do botão “Voltar atrás”, mas se o estudante tiver falhado o desafio final é fornecida a possibilidade de voltar a tentar a quantidade de vezes que necessitar até conseguir introduzir código válido e executável através de um botão criado para esse efeito denominado “Rever Código” como podemos observar na figura 4.13. Com esta possibilidade, o falhanço passa a ser visto como algo favorável, uma vez que o estudante terá

tentativas ilimitadas para ser autodidata ao ponto de ultrapassar um desafio de maior grau e dessa maneira evoluir como programador.

A componente das estatísticas, que aloja o template da página das estatísticas, visível na figura 4.14, é de extrema importância a nível da gamificação uma vez que através da implementação de aspetos visuais fortes e vibrantes proporciona ao estudante uma sensação de imersão num universo de jogo, sobretudo pela existência de uma secção correspondente aos crachás que foram desbloqueados bem como os crachás bloqueados.

Neste Componente existe um método que, através do serviço de autenticação por nós desenvolvido, obtém todas propriedades existentes no nó do estudante que se encontra na base de dados. Em relação aos crachás, é chamado um método do serviço de gamificação que é responsável por obter os dados existentes no nó dos crachás associado ao nó do estudante. Se a propriedade *estáDesbloqueado* do crachá for verdadeira, o ícone referente a esse mesmo crachá será clicável, permitindo a sua visualização numa modal como é observável através da figura 4.15, e não terá opacidade fornecendo um aspeto mais vivo ao ecrã que irá ter impactos psicológicos positivos no estudante que se sentirá motivado em prosseguir com a aventura uma vez que têm recebido as devidas recompensas pelo seu trabalho. Todas estas propriedades relativas ao estudante e aos seus crachás encontram-se sempre sincronizadas com os dados existentes na base dados e facilitam a visualização do progresso do estudante, fator de extrema importância a nível educacional, uma vez que existe a necessidade de fornecimento de feedback em todas as etapas do ensino.

Como acréscimo da gamificação e pelo fato de ser importante que o estudante se sinta verdadeiramente realizado, sempre que se entra nesta página é verificado quantos crachás o estudante já conquistou, e caso falte um para completar a coleção, é o desbloqueado o último crachá denominado “Aventureiro”. Esta conquista indica ao estudante que a sua experiência no protótipo terminou com sucesso uma vez que foi conquistado tudo o que havia para conquistar, fazendo com que haja um contributo e impacto real na personalidade do estudante, uma vez que o mesmo se irá sentir realizado e consequentemente motivado em continuar com a aprendizagem da programação para além do protótipo.

Por fim, a componente da página tabela classificativa vai ao encontro do princípio de relacionamento da teoria da Auto-Determinação proposta por Deci e Ryan (1985) uma vez que nesta página o estudante poderá verificar em que medida as suas conquistas se relacionam com

as dos seus colegas de curso, promovendo o espírito competitivo do mesmo. Assim que se entra nesta página é efetuada uma chamada ao método do serviço de gamificação responsável por obter todos os nós dos estudantes presentes na base de dados que se encontrem no modo gamificado ordenados por pontuação total, como podemos observar na figura 4.17.

4.3.3.3 Diagrama de Componentes

Na figura 4.46 é possível ter uma visão geral da arquitetura do protótipo através de um diagrama de componentes.

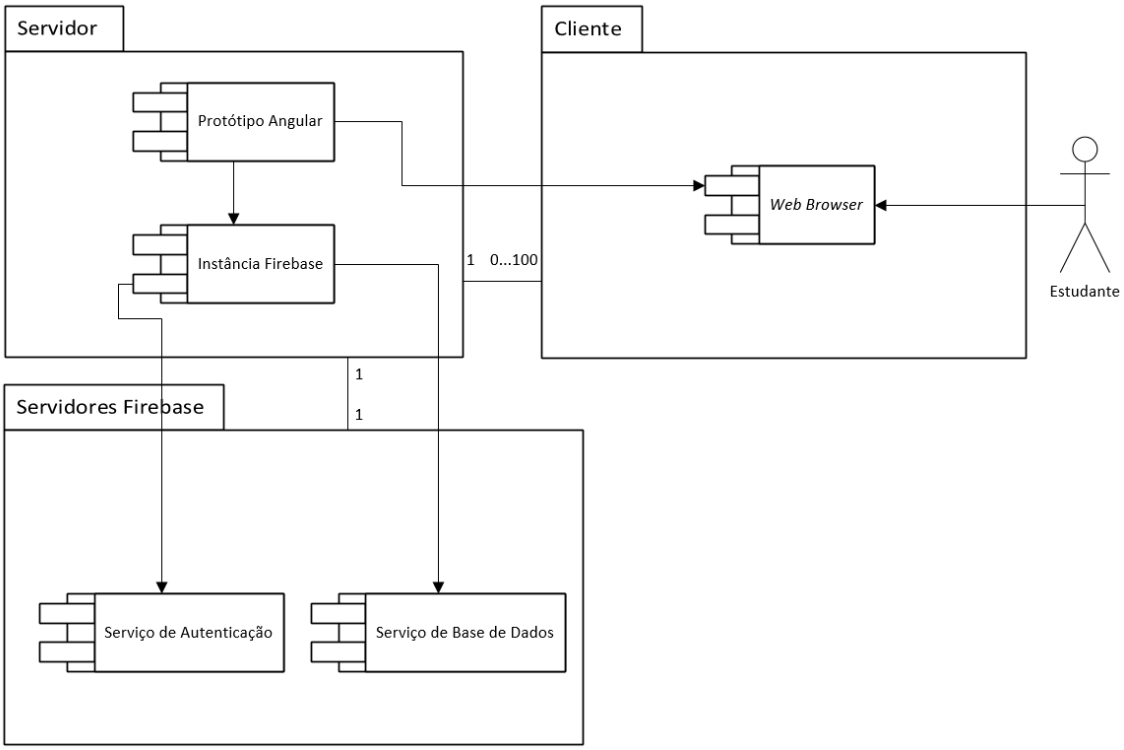


Figura 4.46: Diagrama de Componentes

A implementação deste protótipo segue a tradicional arquitetura Cliente-Servidor, com a vantagem de existirem servidores geridos pela empresa Firebase que auxiliam na gestão de autenticações e de base de dados.

O componente servidor pode ser visto como o principal uma vez que é composto pelo subcomponente do protótipo desenvolvido em angular anteriormente detalhado, bem como a subcomponente da instância do Firebase que permitirá o acesso aos serviços de autenticação e serviços de base de dados disponibilizados pelos servidores do Firebase. Estes dois

subcomponentes interligam-se através de uma chave API, disponibilizada pela plataforma de desenvolvimento do Firebase.

O componente cliente é apenas composto pelo subcomponente *web browser* que por sua vez é o único elemento com o qual o estudante interagirá podendo ser visto como a ponte de ligação entre o estudante e os restantes componentes do protótipo.

O componente servidores Firebase é vital para o funcionamento do protótipo, pois aloja o subcomponente serviços de autenticação, que contém todos os registos e *tokens* de autenticação dos estudantes, bem como o subcomponente serviços de base de dados que contém todos os dados relativos às questões, respostas e dados dos estudantes que alimentarão o protótipo angular. A comunicação destes subcomponentes com o componente servidor é efetuada através da subcomponente instância Firebase, que serve como intermediário entre o protótipo e os serviços do Firebase.

Existe apenas uma instância do componente servidor que por sua vez pode comunicar com até cem instâncias do componente cliente, uma vez que foi utilizada a versão free do Firebase, e com uma instância do componente dos servidores do Firebase. Esta última comunicação é essencial para garantir que todas as funcionalidades do subcomponente protótipo funcionem conforme esperado.

5. Resultados

Neste capítulo serão apresentados e analisados os dados recolhidos aquando a utilização do protótipo implementado. O período de execução deste estudo ocorreu desde dia dezasseis de Junho de Dois mil e Dezassete até dia vinte e um de Julho de dois mil e dezassete, com uma duração total de 35 dias. Este capítulo corresponde à quinta e sexta fases da abordagem metodológica da dissertação: avaliação do artefacto e comunicação.

5.1 Avaliação do artefacto

Numa primeira instância foi fornecido o *url* do *website* desenvolvido através do facebook para que os curiosos pudessem participar neste estudo tendo sempre em conta o livre arbítrio de cada individuo.

Através do gráfico 5.1 podemos verificar que a amostra ficou distribuída equitativamente uma vez que existem vinte e quatro pessoas inscritas na vertente não gamificada do protótipo e vinte e quatro pessoas inscritas na vertente gamificada do protótipo.

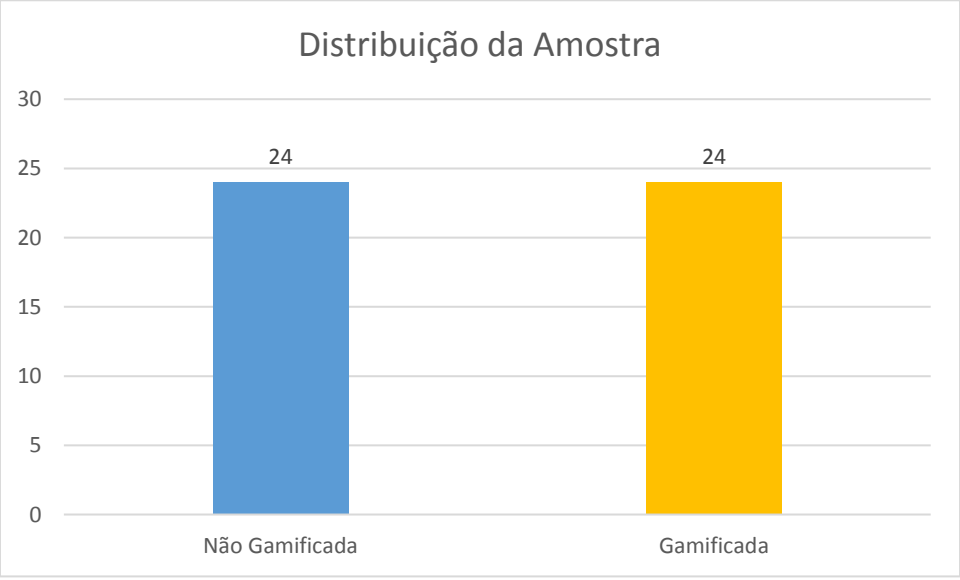


Gráfico 5.1: Distribuição da Amostra

Uma vez que o foco deste estudo se prende com a obtenção de uma resposta acerca do problema inicialmente identificado é importante analisarmos os dados relativos à utilização do protótipo

para se perceber se houve diferenças ao nível das motivações dos estudantes perante as duas vertentes existentes.

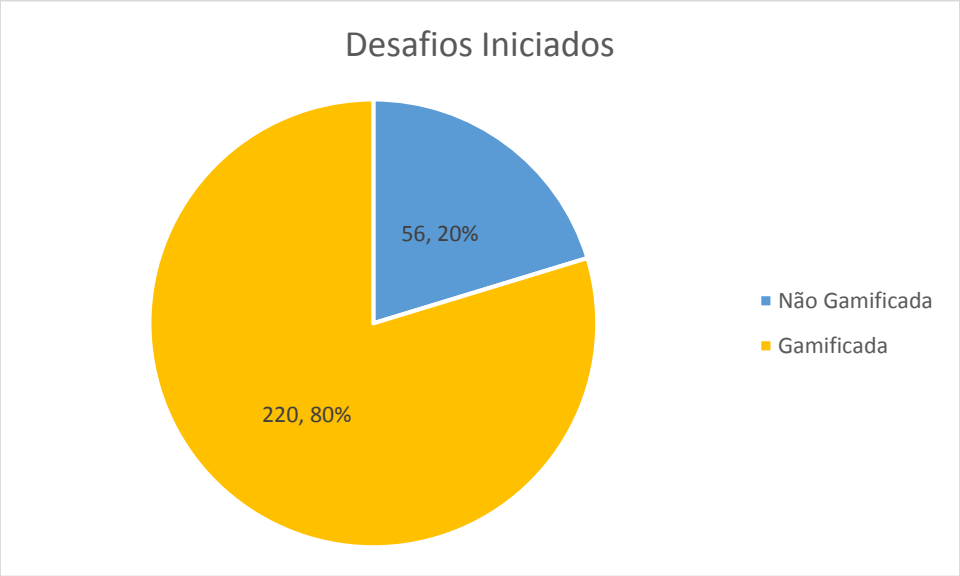


Gráfico 5.2: Desafios Iniciados

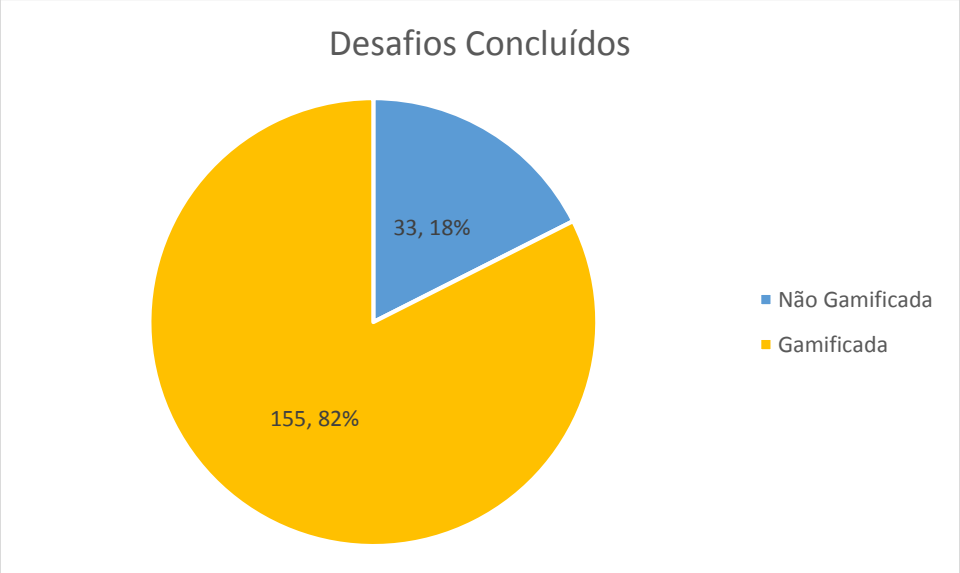
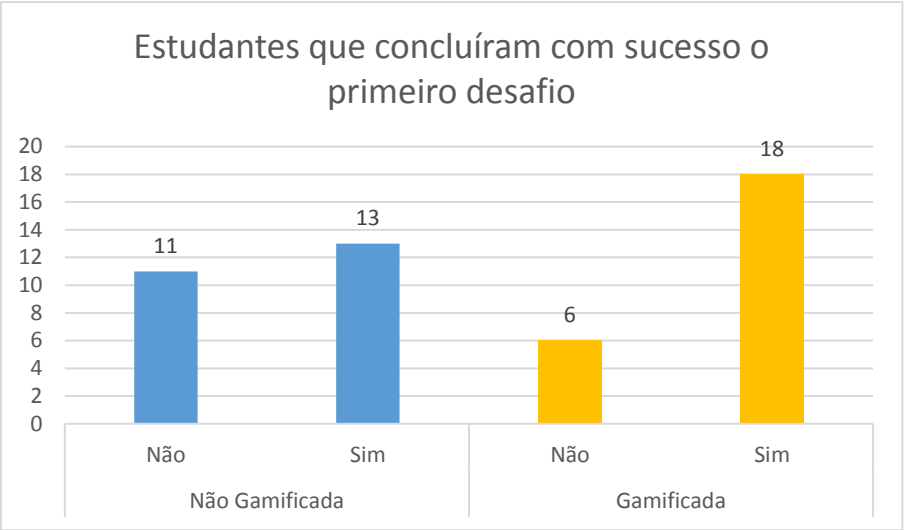


Gráfico 5.3: Desafios Concluídos

Como podemos observar através dos gráficos 5.2 e 5.3, houve uma grande diferença em relação à quantidade de desafios realizados em cada uma das vertentes. Na vertente não gamificada apenas foram iniciados 56 desafios e concluídos 33, ou seja, a percentagem de conclusão foi de 59%. Na vertente gamificada, foram iniciados quatro vezes mais desafios sendo que em 220 desafios, 155 foram concluídos, correspondendo a uma percentagem de conclusão de 70%. As

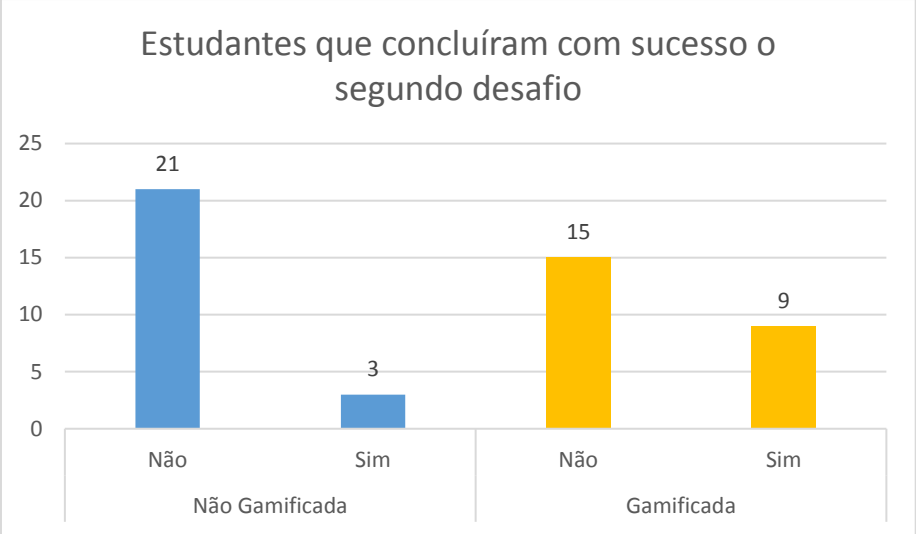
conclusões dos desafios tiveram em conta os desafios concluídos com sucesso e insucesso, mas podemos verificar que houve um aumento de 11% ao nível das suas conclusões da vertente gamificada em relação à vertente não gamificada. Este facto pode indicar que os estudantes se sentiam mais motivados, empenhados e interessados em realizar tarefas relativas à aprendizagem de programação quando existia a presença de gamificação.



Nota: Não – Número de estudantes que não concluíram o primeiro desafio com sucesso
Sim - Número de estudantes que concluíram o primeiro desafio com sucesso

Gráfico 5.4: Estudantes que concluíram com sucesso o primeiro desafio

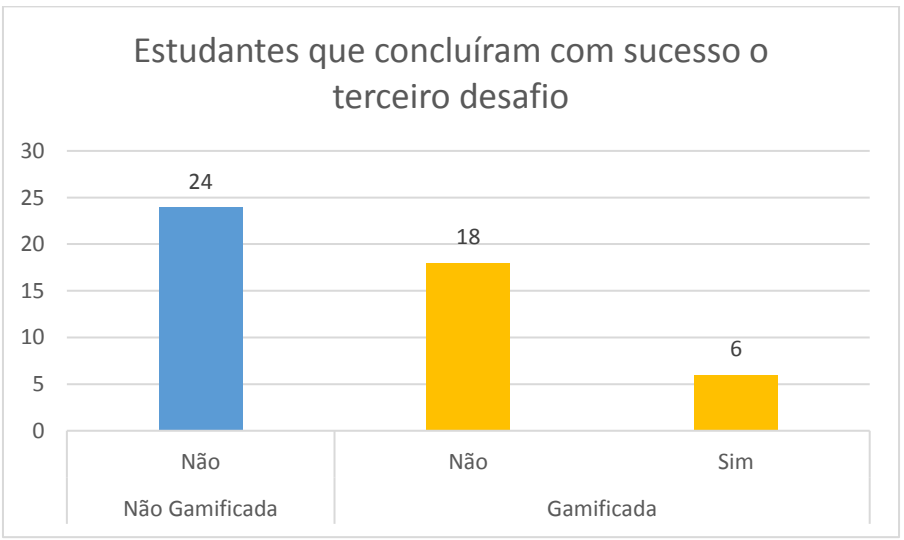
Verificou-se que existiram diferenças significativas relativamente às conclusões dos três desafios propostos. Através da análise do gráfico 5.4, constata-se que 13 estudantes ultrapassaram com sucesso o primeiro desafio na vertente não gamificada, enquanto na vertente gamificada 18 estudantes conseguiram esse mesmo feito. Na vertente não gamificada 54% dos estudantes ultrapassaram o primeiro desafio, enquanto na vertente gamificada este número foi de 75%. Existe portanto uma diferença de 39% em relação à conclusão do primeiro desafio nas duas vertentes.



Nota: Não – Número de estudantes que não concluíram o segundo desafio com sucesso
Sim - Número de estudantes que concluíram o segundo desafio com sucesso

Gráfico 5.5: Estudantes que concluíram com sucesso o segundo desafio

Em relação ao segundo desafio, e como podemos observar no gráfico 5.5, na vertente não gamificada apenas 3 estudantes o concluíram com sucesso, existindo uma taxa de conclusão de apenas 13%. Por outro lado, na vertente gamificada 9 estudantes concluíram com sucesso o segundo desafio, o que corresponde a uma taxa de conclusão de 38%. Apesar de ambas as percentagens serem baixas, vericou-se que houve um aumento de 200% de conclusões da vertente gamificada em relação à vertente não gamificada.



Nota: Não – Número de estudantes que não concluíram o terceiro desafio com sucesso
Sim - Número de estudantes que concluíram o terceiro desafio com sucesso

Gráfico 5.6: Estudantes que concluíram com sucesso o terceiro desafio

Como é possível observar através do gráfico 5.6, na vertente não gamificada nenhum estudante conseguiu concluir o terceiro desafio enquanto na vertente gamificada apenas 6 estudantes conseguiram este feito. A taxa de conclusão do terceiro desafio na vertente não gamificada foi de 0% enquanto na vertente gamificada foi apenas 25%.

Podemos concluir que apesar das baixas taxas de conclusão do segundo e terceiro desafio em ambas as vertentes, houve uma ligeira melhoria nas taxas de conclusão dos desafios na vertente gamificada. Este facto poderá indicar que os estudantes se sentiram demasiado desafiados aquando a realização dos desafios propostos, acabando por desmotivar e não se empenharem o suficiente para os concluírem. No entanto parece que a presença de gamificação teve impacto positivo, uma vez que os estudantes na vertente gamificada sentiram-se mais motivados a alcançar melhores resultados e a continuar a utilizar o protótipo gamificado, o que pode ser explicado através dos números presentes nos três gráficos anteriores.

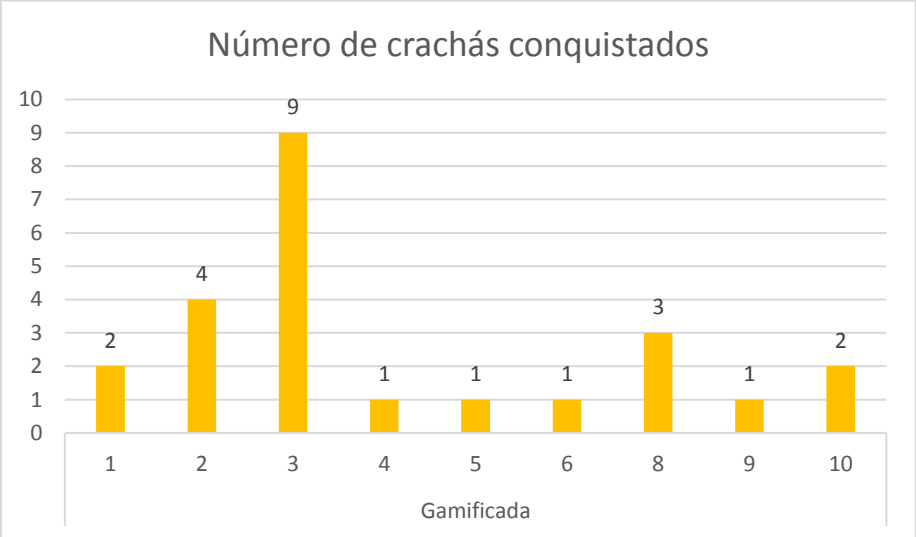


Gráfico 5.7: Número de crachás conquistados

Não foi possível concluir se houve impacto direto nas motivações dos estudantes relativamente à existência de crachás na vertente gamificada, uma vez que 63% dos estudantes apenas conquistaram até 3 crachás que corresponde à conclusão do primeiro desafio. Os restantes 37% conseguiram conquistar mais crachás mas apenas 25% dos estudantes conseguiram obter 8 ou mais crachás.

Gráfico 5.8: Níveis alcançados

Em relação à existência de níveis também não parece ter havido grande impacto nas motivações dos estudantes uma vez que 38% dos estudantes não passou do primeiro nível e apenas 29% dos estudantes alcançou o nível máximo.

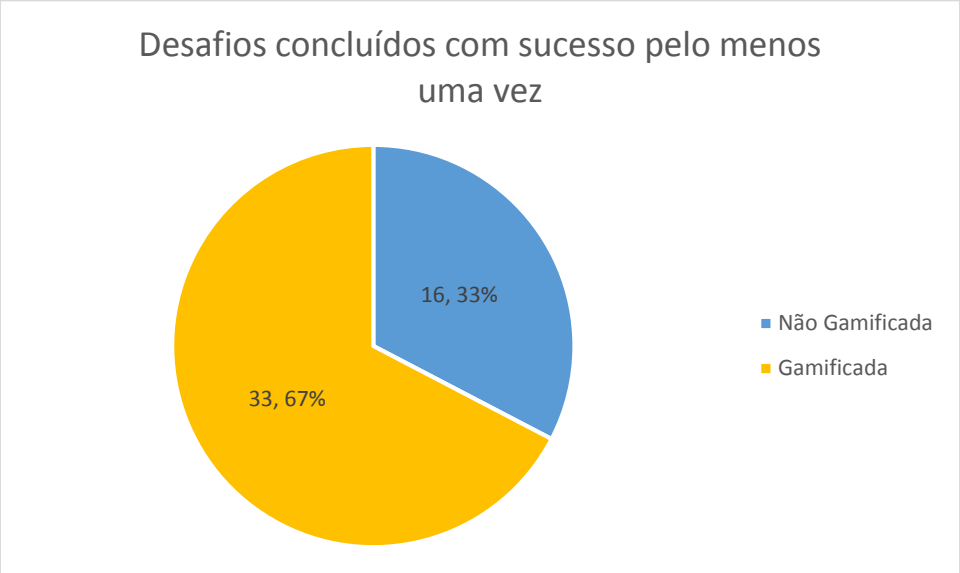


Gráfico 5.9: Desafios concluídos com sucesso pelo menos uma vez

O gráfico 5.9, evidência que pode existir uma diferença significativa nas motivações dos estudantes quando é implementada gamificação. Este gráfico diferencia-se do gráfico 5.3 uma vez que com a implementação da gamificação é desejável que os estudantes possam repetir inúmeras vezes os desafios de modo a absorverem cognitivamente todos os conceitos de

programação disponibilizados pelo protótipo. No entanto o gráfico 5.9 apenas se refere às primeiras conclusões dos desafios, que por sua vez desbloqueariam os desafios seguintes na vertente gamificada.

Em ambas as vertentes, existiam 24 estudantes que tinham a possibilidade de concluir 3 desafios cada, resultando num total de 72 a serem concluídos em cada uma das vertentes. Na vertente não gamificada apenas 16 desafios foram concluídos, resultando numa taxa de sucesso de 22%, enquanto na vertente gamificada foram concluídos 33 desafios correspondendo a uma taxa de sucesso de 46%, ou seja, o dobro relativamente à taxa de sucesso da vertente não gamificada.

Realizámos ainda um teste t de modo a saber se a hipótese nula "O uso da Gamificação não têm influência no aumento da motivação dos estudantes na aprendizagem de programação" deveria ou não ser rejeitada. Para o efeito usamos os dados relativos ao gráfico 5.9

Tabela 5.1: Teste t para igualdade de médias.

	Não Gamificada	*Gamificada*
Média	0,67	1,38
Variância	0,49	1,29
Observações	24,00	24,00
Correlação (Pearson)	0,11	
Hipótese de Igualdade de Médias	0,00	
Graus de Liberdade	23,00	
Estatística t	-2,74	
P(T<=t) uma cauda	0,01	
t Critico uma cauda	1,71	
P(T<=t) duas caudas	0,01	
t Critico duas caudas	2,07	

Analisando a tabela 5.1 podemos verificar que existe uma diferença significativa em relação à média de desafios concluídos com sucesso pelo menos uma vez. Na vertente não gamificada os estudantes em média não chegam a concluir 1 desafio, sendo a média 0,67 desafios concluídos, enquanto na vertente gamificada a média aumenta para 1,38 demonstrando que os estudantes quando em contato com a gamificação concluíram pelo menos 1 desafio.

A correlação de Pearson tem como valor 0,11 e portanto significa que não existe uma grande associação entre as duas vertentes.

O valor probabilístico é 1% o que significa que podemos rejeitar a hipótese nula referida anteriormente com um nível de certeza de 99%, o que nos indica que a gamificação parece ter efetivamente impacto na motivação dos estudantes perante a aprendizagem da programação.

5.2 Comunicação

Nesta fase da abordagem metodológica descreve-se os momentos de comunicação dos resultados desta dissertação. Parte desta dissertação foi publicada numa conferencia internacional em forma de comunicação cientifica (Pereira, Costa & Aparício, 2017), nomeadamente na conferencia CISTI 2017, indexada na Scopus, ISI Web of Knowledge e publicada na IEEE.

Os resultados desta dissertação e nomeadamente o protótipo foi testado por dezenas de participantes no evento da Noite Europeia dos Investigadores (NEI, 2017). Tendo sido comunicado os principais resultados desta investigação e demonstrado o uso da plataforma gamificada. (Figuras 5.1; 5.2; 5.3; 5.4 e 5.5)

Figura 5.1: Noite dos Investigadores 2017

Figura 5.2: Noite dos Investigadores 2017, demonstração da plataforma gamificada

Figura 5.3: Noite dos Investigadores 2017, demonstração da plataforma gamificada a participantes do ensino básico no NEI 2017

Figura 5.4: Noite dos Investigadores 2017, uso e demonstração da plataforma gamificada

Figura 5.5: Noite dos Investigadores 2017, Carlos J. Costa (Orientador) e Ricardo Pereira (Orientando)

6. Conclusões e Trabalho Futuro

6.1. Conclusões

Esta dissertação foi realizada com o intuito de analisar que medida a gamificação contribui para melhorar a aprendizagem da programação.

O primeiro passo dado foi a realização do estado de arte onde foi possível constatar que existem imensos desafios na aprendizagem relativamente a unidades curriculares que envolvam conceitos de programação. Um dos principais e talvez o maior desafio a enfrentar é a falta de motivação que os estudantes apresentam perante estas matérias. Apesar de terem sido efetuados diversos esforços por parte de professores e investigadores no sentido de desenvolver soluções para colmatar este problema, ainda se levantam muitas dúvidas quanto à sua eficácia no ensino da programação. É neste contexto que a gamificação surgiu como uma possível.

Tentando obter uma solução que combatesse este problema recorrente entre os estudantes propusemos um modelo conceptual no qual colocamos em primeiro plano a gamificação como principal potenciador das motivações dos estudantes aquando a utilização de sistemas de aprendizagem de programação.

No seguimento do modelo proposto desenvolvemos um protótipo recorrendo a tecnologias *open-source* que permitissem aferir se efetivamente a gamificação poderia ter impactos positivos nas motivações dos estudantes.

Após distribuição e utilização do protótipo por parte da amostra, verificou-se através da análise dos resultados obtidos que efetivamente a gamificação pode ter um impacto bastante positivo nas motivações dos estudantes quando estamos perante tópicos de programação. Esta afirmação, no entanto, carece de um estudo mais profundo uma vez que a amostra foi relativamente pequena quando abordamos uma temática de extrema importância como esta. Tendo sido ainda testado numa fase posterior em ambiente publico no evento da Noite Europeia dos Investigadores de 2017.

6.2. Limitações e Trabalhos Futuros

Fazendo uma introspeção do que foi alcançado nesta dissertação e no que poderá ser feito para melhorar esta temática propomos que os seguintes tópicos sejam abordados e melhorados:

- Em primeiro lugar achamos que o protótipo pode ser melhorado uma vez que a gamificação disponibiliza infinitas possibilidades de implementação. Propomos, por exemplo, a implementação de elementos sonoros de modo fomentar uma maior absorção cognitiva por parte dos estudantes.
- Propomos que seja realizado um estudo onde seja implementado um protótipo semelhante em diversos ambientes escolares não ficando apenas cingido ao ensino universitário. Seria interessante realizar um estudo onde se comparasse os resultados da implementação de um protótipo gamificado no primeiro semestre do ensino universitário contra os resultados obtidos da implementação do mesmo protótipo no segundo período do ensino secundário, por exemplo.
- Achamos que um protótipo deste género traria bastantes benefícios se fosse implementado como elemento de avaliação extra de uma unidade curricular de programação. Seria interessante realizar um estudo acerca do impacto desta implementação nas notas dos estudantes.

Referências

Agarwal, R. and Karahanna, E. (2000) Time Flies When You're Having Fun: Cognitive Absorption and Beliefs about Information Technology Usage. Journal MIS Quarterly, Vol. 24, No. 4, pp. 665–694.

Ala-Mutka, K. (2004). Problems in learning and teaching programming–a literature study for developing visualizations in the Codewitz-Minerva project. *Codewitz needs analysis*, 1-13. Available: http://www.cs.tut.fi/~edge/literature_study.pdf.

Antunes, P., & Costa, C. J. (2002). Handheld CSCW in the meeting environment. In *International Conference on Collaboration and Technology* (pp. 47-60). Springer, Berlin, Heidelberg

Barata, G., Gama, S., Jorge, J. and Gonçalves, D. (2013) Engaging engineering students with gamification. In 2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), pp. 1–8.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives, handbook I: The cognitive domain* (Vol. 19, p. 56). New York: David McKay Co Inc.

Caponetto, I., Earp, J. and Ott, M. (2014) Gamification and education: A literature review. In ECGBL 2014: Eighth European Conference on Games Based Learning, pp. 50–57.

Caspersen, M. E. and Bennedsen, J. (2007) Instructional design of a programming course: A learning theoretic approach. In Proceedings of the Third International Computing Education Research Workshop (ICER 2007), Atlanta, GA, pp. 111–122.

CombéFis, S., BeresnevičIus, G. and Dagienė, V. (2016) Learning Programming through Games and Contests: Overview, Characterisation and Discussion. Olympiads in Informatics, vol. 10, no. 1, pp. 39–60.

Costa, C. J., & Aparício, M. (2006). Computer Game–Discussing Development Process. In *Proceedings of IRIS* (Vol. 29).

Costa, C. J., & Aparicio, M. (2014). Evaluating success of a programming learning tool. In *Proceedings of the International Conference on Information Systems and Design of Communication* (pp. 73-78). ACM.

Costa, C. Aparicio, M. and Cordeiro, C. (2012 a). A solution to support student learning of programming. In Proceedings of the Workshop on Open Source and Design of Communication (OSDOC '12). ACM, New York, NY, USA, 25-29.

Costa, C. J., Aparicio, M., & Cordeiro, C. (2012 b). Web-Based graphic environment to support programming in the beginning learning process. In *International Conference on Entertainment Computing* (pp. 413-416). Springer, Berlin, Heidelberg.

Costa, C. J., Aparicio, M., & Pierce, R. (2009). Evaluating information sources for computer programming learning and problem solving. In *Proceedings of the 9th WSEAS International Conference On Applied Computer Science* (pp. 218-223).

Csikszentmihalyi, M. (1990). Flow. The Psychology of Optimal Experience. New York (HarperPerennial) 1990.

Deci, E. and Ryan, R. M. (1985) Intrinsic Motivation and Self-Determination in Human Behavior. New York: Plenum Press.

Deterding, S., Dixon, D., Khaled, R. and Nacke, L. (2011) From game design elements to gamefulness: defining gamification. Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments, pp 9-15.

Deterding, S., Khaled, R., Nacke, L. and Dixon, D. (2011) Gamification: Toward a definition. In CHI 2011 gamification workshop.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K. and Dixon, D. (2011) Gamification. using game-design elements in non-gaming contexts. In CHI 2011 Extended Abstracts on Human Factors in Computing Systems, pp. 2425–2428.

Dicheva, D., Dichev, C., Agre, G. and Angelova, G. (2015) Gamification in education: a systematic mapping study. Journal of Educational Technology & Society, Vol. 18, No. 3 (July 2015), pp. 75-88.

Groh, F. (2012) Gamification: State of the art definition and utilization. In Proceedings of the 4th Seminar on Research Trends in Media Informatics, pp. 39-46.

Hamari, J., Koivisto, J. and Sarsa, H. (2014) Does gamification work?–a literature review of empirical studies on gamification. In 2014 47th Hawaii International Conference on System Sciences, pp. 3025–3034.

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9-22). Springer US.

Ibáñez, M.-B., Di-Serio, A. and Delgado-Kloos, C. (2014) Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. IEEE Transactions on Learning Technologies, Vol. 7, No. 3, pp. 291–301.

Jarvinen, P. (2000). Research Questions Guiding Selection of an Appropriate Research Method. (pp. 124–131). Presented at the ECIS.

Iosup, A. and Epema, D. (2014) An experience report on using gamification in technical higher education. Proceedings of the 45th ACM technical symposium on Computer science education, pp. 27–32.

Khaleel, F. L., Ashaari, N. S., Meriam, T. S., Wook, T. and Ismail, A. (2015) The study of gamification application architecture for programming language course. Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, pp. 1–5.

Knutas, A., Ikonen, J., Nikula, U. and Porras, J. (2014) Increasing collaborative communications in a programming course with gamification: a case study. Proceedings of the 15th International Conference on Computer Systems and Technologies, pp. 370–377.

Koivisto, J. and Hamari, J. (2014) Demographic differences in perceived benefits from gamification. Computers in Human Behavior, Vol. 35, pp. 179–188.

Koster, R. (2013) Theory of Fun for Game Design. O'Reilly Media, Inc.

Lee, J. J. and Hammer, J. (2011) Gamification in education: What, how, why bother? Academic Exchange Quarterly, Vol. 15, No. 2, pp. 1–5.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266. https://doi.org/10.1016/0167-9236(94)00041-2

Marczewski, A. (2013). *Gamification: a simple introduction*. Andrzej Marczewski.

NEI. (2017). Programas - Noite Europeia dos Investigadores [2017]. Retrieved September 30, 2017, from http://noitedosinvestigadores.pt/acontece/programas/

Oliver, R. and Herrington, J. (2003) Exploring Technology-Mediated Learning from a Pedagogical Perspective. Interactive Learning Environments, Vol. 11, No. 2, pp. 111–126.

Olsson, M., Mozelius, P. and Collin, J. (2015) Visualisation and Gamification of e-Learning and Programming Education. The Electronic Journal of e-Learning, Vol. 13, No. 6, pp. 441-454.

Pereira, R. Costa, C. and Aparicio. J. (2017) Gamification to support programming learning. In 2017 12th Iberian Conference on Information Systems and Technologies (CISTI). Lisbon, Portugal: IEEE.

Pierce, R., & Aparício, M. (2010). Resources to support computer programming learning and computer science problem solving. In *Proceedings of the Workshop on Open Source and Design of Communication* (pp. 35-40). ACM.

Piteira, M. and Costa, C. (2012). Computer programming and novice programmers. In Proceedings of the Workshop on Information Systems and Design of Communication (ISDOC '12). ACM, New York, NY, USA, 51-53.

Piteira, M. and Costa, C. (2017) Gamification: Conceptual framework to online courses of learning computer programming. In 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1-7.

Piteira, M. and Costa. C. (2013). Learning computer programming: study of difficulties in learning programming. In Proceedings of the 2013 International Conference on Information Systems and Design of Communication (ISDOC '13). ACM, New York, NY, USA, 75-80.

Piteira, M. Costa, C. and Haddad, C. (2012). Educational computer programming tools. In Proceedings of the Workshop on Open Source and Design of Communication (OSDOC '12). ACM, New York, NY, USA, 57-60.

Rodrigues, L. F., Costa, C. J., & Oliveira, A. (2013). How to develop financial applications with game features in e-banking?. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication* (pp. 124-134). ACM

Rodrigues, L. F., Costa, C. J., & Oliveira, A. (2014). How gamification can influence the web design and the customer to use the e-banking systems. In *Proceedings of the*

International Conference on Information Systems and Design of Communication (pp. 35-44). ACM.

Rodrigues, L. F., Oliveira, A., & Costa, C. J. (2016 a). Playing seriously–How gamification and social cues influence bank customers to use gamified e-business applications. *Computers in Human Behavior*, *63*, 392-407.

Rodrigues, L. F., Oliveira, A., & Costa, C. J. (2016 b). Does ease-of-use contributes to the perception of enjoyment? A case of gamification in e-banking. *Computers in Human Behavior*, *61*, 114-126

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P. and Queirós, R. (2012) A distributed system for learning programming on-line. Journal Computers & Education, Vol. 58, No. 1, pp. 1–10.

Vihavainen, A., Airaksinen, J. and Watson, C. (2014) A systematic review of approaches for teaching introductory programming and their influence on success. In Proceedings of the tenth annual conference on International computing education research, pp. 19–26.