



Department of Information Science and Technology

**From Native to Cross-platform Hybrid Development:
CodeGT - Design and development of a mobile app for an ERP**

Carlos Manuel Manso Pinto

Dissertation submitted as partial fulfilment of the requirements for the degree of
Master in Computer Engineering

Supervisor:
Professor Carlos Eduardo Dias Coutinho, Assistant Professor, ISCTE-IUL

October 2018

Acknowledgements

I would like to thank my supervisor, Professor Carlos Eduardo Dias Coutinho, thank you for your guidance, knowledge, availability and time.

Many thanks to my family, the lack in the retreat periods and to understanding the goal to be achieved.

To my ISCTE-IUL Teachers, for the learning skills.

To Sigecom – IT Solutions Lda, my company for the support and time for the preparation of the thesis.

To all that I have listed my sincere "Thank you".

Resumo

As tendências atuais em direção à grande mobilidade dos indivíduos, juntamente com o crescimento exponencial do número de dispositivos móveis, levaram ao enorme crescimento na procura do desenvolvimento de aplicações móveis. Além disso, com a expansão e heterogeneidade dos dispositivos e das plataformas móveis, as empresas de desenvolvimento de software necessitam de encontrar formas mais rápidas e baratas de desenvolver aplicações capazes de abranger o maior número de dispositivos para ir ao encontro da elevada procura do mercado. Atualmente, os sistemas operativos Android e iOS dividem e dominam o mercado de dispositivos móveis com expressões tímidas de outros concorrentes. Cada um desses sistemas operativos móveis foi desenvolvido especificamente para linguagens de programação e estratégias próprias e oferecem um conjunto de ferramentas de desenvolvimento com as suas bibliotecas, para a criação de aplicações nativas. Por outro lado, a evolução do HTML5, CSS e do JavaScript criaram oportunidades para o surgimento de alternativas genéricas para criação de aplicações multiplataforma que correm em todos os dispositivos e em todos os sistemas operativos, mas sem a capacidade de aceder todo o potencial nativo do dispositivo. Paralelamente surgiram as novas plataformas de desenvolvimento híbridas, que tentam tirar o melhor partido dos dois mundos.

Esta dissertação descreve a evolução das diferentes abordagens no desenvolvimento de aplicações móveis mais concretamente na utilização de ferramentas multiplataformas para a criação de aplicações móveis híbridas e as suas vantagens. A pesquisa incluiu ainda o desenvolvimento de uma aplicação móvel, CodeGT, desenvolvido numa plataforma híbrida para interagir com um *software* ERP, acedendo aos Documentos de Transporte registados nesse ERP, assim como ao código transmitido pela Autoridade Tributária (AT), que assim dispensa a impressão de documentos e indo ao encontro de uma necessidade do mercado. Esta aplicação já tem empresas clientes interessadas nela.

Palavras-Chave: desenvolvimento de aplicações; aplicação móvel; aplicações móveis; Ionic; plataforma Ionic; plataformas desenvolvimento híbrido; desenvolvimento de aplicações multiplataforma; desenvolvimento de aplicações híbridas; plataformas desenvolvimento móveis; aplicações híbridas; Enterprise Resource Planning; software ERP; multi-plataforma; Angular.

Abstract

The current trend towards mobility of individuals, together with the exponential growth of the number of mobile devices led the market to a boom in the demand for the development of mobile applications. Moreover, with the expansion and heterogeneity of the mobile devices and platforms, software companies need to search for faster and cheaper ways to develop applications that can span as many devices as possible to capture the market. Currently, the Android and iOS Operating Systems roughly share and dominate the mobile market, with timid expressions of other competitors. Each of these mobile operating systems were developed using their own languages, strategy and SDKs for development of applications using their libraries – known as Native apps. On the other hand, the evolution of HTML5, CSS and JavaScript created generic alternatives to create mobile apps that run on devices on all operating systems, although lacking the capability to access the device's full potential. Alongside came the new Hybrid cross-platform development frameworks, which try to take the best of both worlds.

This dissertation describes the evolution of the different mobile app development approaches and the state-of-the-art in their development techniques, and compares them with the Hybrid app approach, then highlighting the trends in mobile app development using Hybrid platforms and their advantages. This research includes the development of a mobile Hybrid application, CodeGT, which interacts with an Enterprise Resource Planning (ERP) to access the Transport Documents registered in this ERP and access to the code transmitted by the Portuguese Tax Authority (AT), therefore not requiring the printing of documents and meeting a need of the business market. This application does already have customer industry companies interested in it.

Keywords: app development; mobile; Ionic; cross-platform; mobile app; Native development; Hybrid development; Native Hybrid app; Enterprise Resource Planning; ERP; multi-platform development; cross-platform development; Hybrid app; Angular; Firebase.

Index

Acknowledgements	i
Resumo	ii
Abstract	iii
Index	iv
List of Tables	vi
List of Figures	vii
Abbreviations	ix
Chapter 1 – Introduction	1
1.1. Motivation.....	2
1.2. Research objectives.....	2
1.3. Research questions.....	3
1.4. Methodology	4
1.5. Structure and organization of the dissertation	4
Chapter 2 – Literature Review	6
2.1. Information Systems	6
2.2. Enterprise Resource Planning (ERP).....	7
2.2.1. Processes of the business areas.....	10
2.2.2. ERP Architectures and Technologies	11
2.2.3. Advantages and disadvantages of an ERP.....	12
2.2.4. Implementing an ERP.....	14
2.2.5. Critical Success Factors.....	15
2.2.6. Maturity and future of ERP systems	15
2.3. Mobile.....	16
2.4. Mobile Applications	19
2.5. Mobile Application Development Tools	21
2.6. The Native Approach.....	22
2.7. The Web Approach.....	23
2.8. The Hybrid Approach	27
2.9. Mobile App Comparison Table: Native vs Web vs Hybrid.....	28
2.10. Mobile Hybrid development	31
2.10.1 Adobe PhoneGap.....	31
2.10.2 Apache Cordova	31
2.10.3 Xamarin Platform	32
2.10.4 Firebase Mobile Platform	33
2.10.5 React.....	34

2.10.6	Meteor.....	35
2.10.7	Ionic Platform.....	35
Chapter 3 – Application Concept and Application Development.....		40
3.1.	Development of the mobile application CodeGT.....	40
3.1.1.	Transport Documents.....	41
3.1.2.	Legal Transport Document Requirement.....	42
3.2.1.	Architecture of Mobile Application.....	44
3.2.2.	Database.....	44
3.2.3.	Server Side.....	45
3.2.4.	Application mockup.....	48
3.3.	Mobile development.....	50
Chapter 4 – Results \ Evaluation.....		69
4.1.	Qualitative results of the prototype.....	69
4.2.	Quantitative Evaluation.....	72
4.2.1	Cost for developing a mobile app.....	72
4.2.1	App Complexity	76
4.2.2	Mobile Hybrid Development Risks	80
4.3.	Analysis and discussion of results.....	81
Chapter 5 – Conclusions.....		82
5.1	Main conclusions.....	82
5.2	Main Scientific and Business Community Contributions.....	84
5.2.1	Contributions at the academic level	84
5.2.2	Contributions at the industry and business level	84
5.3	Future work.....	85
Bibliography.....		86

List of Tables

Table 1 - Native Development Environment per OS	22
Table 2 - Programming languages vs platforms	27
Table 3 - Comparison Native vs web vs Hybrid	30
Table 4 - Evaluation mobile Hybrid app CodeGT	69
Table 5 - SWOT Analysis	70

List of Figures

Figure 1 - Design science research cycles ((Barafort et al., 2018).....	4
Figure 2 - The scope of an ERP (Davenport, 1998)	7
Figure 3 - ERP Evolution	10
Figure 4 - Purchase Process	11
Figure 5 - ERP Architecture	12
Figure 6 - N° Subscribers of mobile land service in Portugal (Portdata, 2018)	16
Figure 7 – Number of Global Computing Users (in millions) Source: comScore, Morgan Stanley 2012	16
Figure 8 - Desktop vs Mobile market share since 2014(in %) (“Mobile Operating System Market Share Worldwide StatCounter Global Stats,” 2018)	17
Figure 9 - Estimate on number of downloaded mobile apps Source: (Statista, 2018b) .	17
Figure 10 - Market share of operating systems in smartphones (in %) (Statista, 2018b)	18
Figure 11 - Mobile Operating System Market Share (in %) (“Mobile Operating System Market Share Worldwide StatCounter Global Stats,” 2018).....	19
Figure 12 - Bootstrap Responsive Web Layout.....	24
Figure 13 - Application architecture for a modern web application Source: (Shahzad, 2017).....	26
Figure 14 - Visual Studio tools for Apache Cordova	32
Figure 15 - C# cross-platform with Xamarin and .Net [Source: (Xamarin, 2018)]	33
Figure 16 - TypeScript is a superset of JavaScript	37
Figure 17 - Hybrid applications architecture and usage of Apache Cordova tools (Bosniac et al., 2017).....	38
Figure 18 - Example of a Transport Document with AT code from ERP PHC	42
Figure 19 - CodeGT Architecture.....	44
Figure 20 - UML from table cl and ft	45
Figure 21 - The MVC pattern	45
Figure 22 - part of Code for GuiasController in C#	48
Figure 23 - Application Mockup	49
Figure 24 - Node.js site	50
Figure 25 - Ionic CLI.....	50
Figure 26 - CLI command to create CodeGT app	51
Figure 27 - Ionic CLI command to start app	51
Figure 28 - Ionic side menu app created.....	51
Figure 29 - Ionic, files structure	52
Figure 30 - src directory	53
Figure 31 - app.html	53
Figure 32 - CLI command to generated page about	54
Figure 33 - New page on Ionic	54
Figure 34 - Html page of Login Page	55
Figure 35 – TypeScript file of Login page	55
Figure 36 - Ionic Stack of Pages.....	56
Figure 37 - Ionic Component - Button	56
Figure 38 - Data model on CodeGT	57
Figure 39 - Firebase Install.....	58
Figure 40 - Firebase Console.....	59
Figure 41 - Provider in Ionic	60
Figure 42 - CLI to build apk.....	60

Figure 43 – CLI to build ipa on a MAC computer	61
Figure 44- Splash screen on iOS, Android and Windows Phone.....	61
Figure 45 – The Welcome and Login Screen	62
Figure 46 - Customers List on iOS, Android and Windows Phone	62
Figure 47 - Layout view from a tablet.....	63
Figure 48 - Customer detail	63
Figure 49 - The Menu screen.....	64
Figure 50 - Pending documents	64
Figure 51 - Detail document.....	65
Figure 52 - Map route to destination	66
Figure 53 - Receiver signature.....	67
Figure 54 - Document delivery and sign	67
Figure 55 - About us screen.....	68
Figure 56 - ERP PHC desktop UI for a specific customer	71
Figure 57 - ERP PHC Web UI for a specific customer	71
Figure 58 - CodeGT UI for a specific customer	72
Figure 59 - Costs on App Development Life Cycle for Single Platform	74
Figure 60 - Costs on App Development Life Cycle for Multi-Platform	75
Figure 61 - Costs on App Development Life Cycle Project for Single Platform.....	75
Figure 62 - Costs on App Development Life Cycle Project for Multi-Platform.....	76
Figure 63 - Apps Complexity Scale	77
Figure 64 - Number of apps available in leading app stores as 1st quarter 2018(Statista, 2018).....	77
Figure 65 - Free vs. Paid Apps (Statista, 2018).....	78
Figure 66 - Share of Global Mobile App Revenue By Type (Appsflyer, 2016)	79
Figure 67 - Number of SMEs in Portugal in 2016 (Pordata, 2016b).....	79
Figure 68 – Percentage of SMEs in Portugal in 2016 (Pordata, 2016a).....	79
Figure 69 - Most pressing long-running challenges facing software developers worldwide as of 2015(Statista, 2015)	80
Figure 70 - Hybrid platforms solution to build mobile apps to interact ERP.....	83

Abbreviations

AJAX - Asynchronous JavaScript and XML

API – Application Programming Interface

APK – Android Package

AT – Autoridade Tributária (Portuguese Government Tax Authority)

BPM – Business Process Management

CLI – Command-Line Interface

CRM – Customer Relationship Management

CRUD - Create, Read, Update, and Delete

CSS – Cascade Style Sheets

DBMS - Data Base Management System

DSR - Design Science Research

EDI – Electronic Data Interchange

ERP – Enterprise Resource Planning

ES - ECMAScript

ESS - Enterprise System Software

GPS – Global Positioning System

HCI - Human-Computer Interaction

HR – Human Resources

HTML – Hypertext Markup Language

IAP – In-App Purchase

ICS - Invent Control Systems

ICT – Information Communication Technologies

IIS – Internet Information Services

IOT – Internet of Things

IPA – iOS App Store Package

IS – Information System

JS – JavaScript

JSON – JavaScript Object Notation

MRP – Material Resource Planning

MVC – Model View Controller

NoSQL – Not Only Structured Query Language

OS – Operating System

PWA – Progressive Web Apps

REST – Representational State Transfer

SASS - Syntactically awesome style sheets

SCM - Supply Chain Management

SDK – Software Development Kit

SMEs – Small and Medium Enterprises

SOAP – Simple Object Access Protocol

SPA – Single Page Application

SQL – Structured Query Language

TCO – Total Cost of Ownership

TTM – Time to Market

UI – User Interface

UML – Unified Modelling Language

UX – User Experience

WCF – Windows Communication Foundation

Chapter 1 – Introduction

The growth of the mobile world has long surpassed the market share of desktop development. The technology innovation in mobile computing, along with the increased capabilities of the mobile devices, the dramatic improvement of the usability and look & feel of portable devices, together with their increasingly cheaper prices has resulted in a massive number of mobile devices in the market. The easy access to this technology and the rapid growth in the numbers of purchased mobile devices resulted in a high demand for mobile applications.

This dissertation describes a brief evolution of the different mobile app development approaches and their state of the art development, and compares them with the Hybrid app development approach, then highlighting the trends in mobile app development using Hybrid platforms and their advantage.

The mobility of individuals and the high growth in the number of mobile devices in recent years has led to the growing need for development of mobile applications. To cover the entire universe of devices, the same application must be developed in several programming languages, consuming a lot of time and money.

Organizations are facing growing competition and market dynamics. Access to information is essential to the efficiency of organization operations. Given the mobility of employees, access to the organization information systems from mobile devices inside and outside the premises is crucial.

Developing mobile applications is currently one of the most important skills a programmer must have. Over the past decade there has been a burst of mobile devices; smartphones, tablets and wearables, which gave rise to a full suite of mobile applications. In a time of mobile applications, how to create them is a difficult challenge for any programmer who needs to know various programming languages to develop for the devices. Languages such as Objective-C, Swift, for the development of iOS devices, Java for Android devices and C# for Windows devices, among others. The emergence of Hybrid development platforms using web development language and technologies, can take advantage of Native device functionality and are interpreted by any mobile device, or operating system emerge as an opportunity for the entire developer community.

Information systems such as Enterprise Resource Planning tools (ERPs) are currently an essential tool in any organization for both legal issues and the need to manage the amount of data organizations have at their disposal. The number of mobile devices has increased year after year and its use has led to the creation of mobile applications to meet the needs of companies in accessing the information available in their ERP and beyond. The mobility of employees and the use of these devices, implies the creation of new architectures of software development appropriate to their screens.

Native versus Hybrid platforms the trend in mobile app development using Hybrid platforms and their advantage. In this dissertation, this approach will be analysed and used a Hybrid development platform to create a mobile application to interact to an ERP.

The mobile Hybrid application developed already has final customers interested in its implementation in their organizations.

1.1. Motivation

Today if companies develop mobile Native apps for all the platforms they need to create several code-base to reach all the device mobile market.

The total time to develop a mobile application is very important, not only for a question of costs, but the need for a quick response to an opportunity or to get an idea on market before the competition, a metric called Time To Market (TTM). Mobile cross-platforms can create mobile apps for several platforms and devices.

There is then a motivation to analyse this cross-platform alternative rather than the Native development.

From a personal point of view, it is a subject that is inextricably linked to my professional experience.

1.2. Research objectives

The purpose of this study is to investigate which platforms and development tools are available in the market to create mobile applications, specifically from the point of view of cross-platforms development.

Development languages of mobile applications are different, and the tendency is for them to continue to diverge. The strategy of the market leaders is different and not going in the sense of a standardization.

1.3. Research questions

To this purpose, the following research questions were raised:

- **RQ1:** Considering the currently large demand for mobile apps, can the development of Hybrid mobile applications be considered an added-value as opposed to developing Native mobile applications?
- **RQ2:** ERPs are complex and heavy software applications, difficult to be accessed from mobile devices, can the development of Hybrid mobile applications be interesting to build interact apps to interface with an ERP?

These research questions will be solved by the confirmation of the following hypotheses, a task which shall be done in the remainder of this document:

- **HYPOTHESIS 1:** The current state is for the existence of multiple and heterogeneous approaches towards the development of mobile applications. These approaches are not converging, they are instead diverging more and more. The approach of develop of Hybrid mobile applications is an added value compared with the Native development from the point of view of cost and effort.
- **HYPOTHESIS 2:** The Hybrid mobile development could be an added value to develop mobile apps to interact, for example, as add-ins to an ERP.

1.4. Methodology

The investigation and research process of this dissertation approaches the Design Science Research (DSR). DSR is inherently a problem-solving process. The DSR methodology focuses on the development of a new artefact (Alan R. Hevner, Salvatore T. March, 2004) (Barafort et al., 2018). An artefact can be represented as a practical solution so that its contribution to the body of knowledge can be supported (Barafort et al., 2018).

The reported research on Process Assessment draws on the DSR methodology for information systems research suggested by Hevner (Hevner, 2007). The DSR methodology, which combines both behavioural and design science paradigms, comprises three interlinked research cycles, relevance, rigour and the central design cycle on Figure 1.

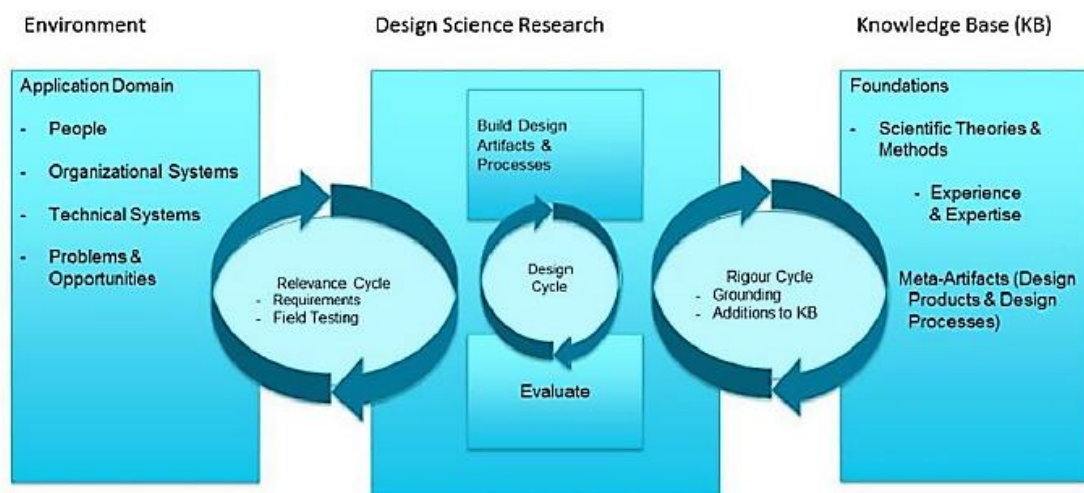


Figure 1 - Design science research cycles ((Barafort et al., 2018)

1.5. Structure and organization of the dissertation

The present study is organized in five chapters that intend to reflect the distinct phases of this dissertation:

- The first chapter introduces the research theme and objectives, research questions and hypothesis, methodology as well as a brief description of the work structure.
- The second chapter represents the theoretical framework, called Literature Review.

- The third chapter is the application of the concept and development of the application.
- The fourth chapter presents the analysis of the results obtained, qualitative and quantitative.
- The fifth and closing chapter presents the conclusions of this study and future work.

Chapter 2 – Literature Review

2.1. Information Systems

The world today lives in the information age. Information is Knowledge and Knowledge is Power. Frequently, businesses mix the concept of information with the concept of storing data. Having a massive load of unstructured and raw data is not sufficient. Information and knowledge only exist upon interpretation of the meaning given to that data. Information management is an essential factor in organizations, in the development of business strategies, in innovation and knowledge, in customer satisfaction, in the improvement of organizational processes (Elbashir, Collier, & Davern, 2008) and in performance measurement.

Information sharing is fundamental in any organization, but some managers remain resistant to giving up their knowledge within organizations, as they still have a vision that only in this way can they maintain a prominent position.

Information systems have emerged in the need to share and access information. Information systems allow the registration and quick access to information in the company.

An information system (IS) registers, stores, processes, analyses and distributes information for a specific purpose (Turban & Volonino, 2011).

Converting the amount of data into information and knowledge is crucial to good decision-making and the success of organizations. Whether for regulatory, legal, market or other reasons, organizations are increasingly confronted with the need to register, obtain, manage and process data generated by organizational processes, thus, the need to use systems of efficient information, thus becoming fundamental.

Globalization, the opening of markets and technological development, imply a great dynamic in the companies. The enormous competitiveness of markets leads to the need to analyse relevant information to decision making as a fundamental success factor in organizations. The access of this information is not only in the workplace but anywhere with mobile devices.

2.2. Enterprise Resource Planning (ERP)

Information systems such as Enterprise Resource Planning (ERP) are currently an essential tool in any organization. ERP is a software used by companies to coordinate all business areas (Koupaei, Mohammadi, & Naderi, 2016). ERP is a solution whose main function is to record the maximum transaction data in different business areas in a single system and in a single relational database (Tomišová & Hudec, 2017) and is a tool that takes part of most companies every day.

The ERP as a computer information system is an application used by companies to coordinate all business areas (Monk & Wagner, 2009), which allows the company to integrate the data of the entire organization (Davenport, 1998), (Chou, Bindu Tripuramallu, & Chou, 2005), into a single central database (Emam, 2013), in a single system (Ganesh, K. , Sivakumar, 2014). The ERP application accesses an application database in a unified interface across the enterprise (Tadger, 1998).

ERP enables companies to have a single integrated and modular software instead of multiple applications that do not communicate with each other. The ERP also avoids the redundancy of records between different databases and allows the efficiency in the processes and the operational analysis of the company. It is thus beyond the difficulty of having in the company, several applications that do not communicate with each other and with duplicate records in different databases. Figure 2 shows the large scope of an ERP (Davenport, 1998).

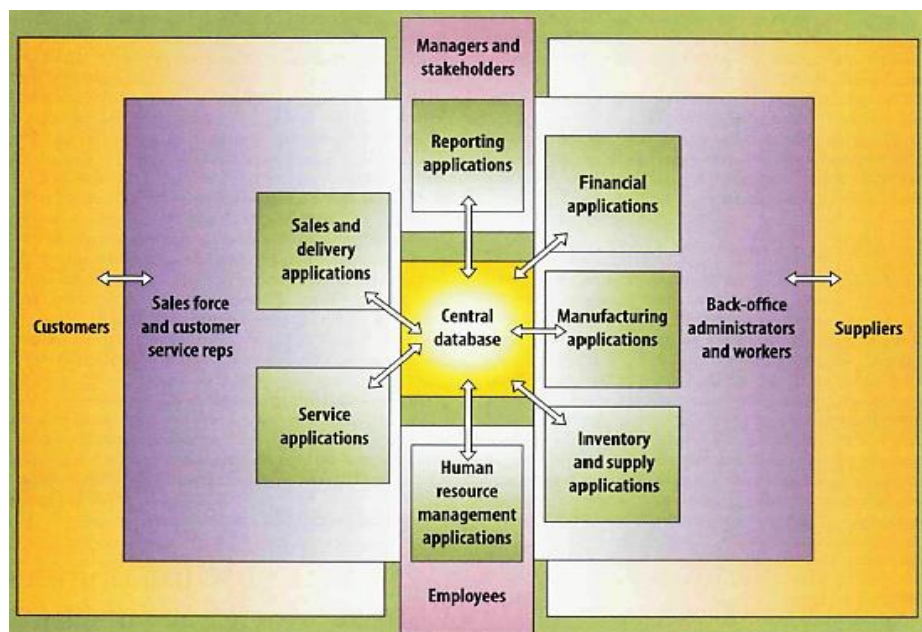


Figure 2 - The scope of an ERP (Davenport, 1998)

The current strategy of software companies that develop and sell ERPs is to make the application available through various modules (Ganesh, K. , Sivakumar, 2014) (Emam, 2013), such as Management, Human Resources, Accounting, CRM, Logistics, Production, Project, Support among others. Organizations can acquire more modules and integrate them throughout the life of the company, according to their financial resources and needs, arising from the development of the company, thus adapting to lower initial investments that can be complemented when there are budgets for more licensing, implementations and process integrations.

These software solutions consist of integrated modules (Ganesh, K. , Sivakumar, 2014) that can be added at any time. ERP enables companies to have a single integrated, modular software instead of multiple applications. When referring to ERP the most important element are the existing processes in organizations. Processes that must be mirrored in the integrated application and which are recorded in a relational database, Data Base Management System (DBMS).

Until 2000 it was common for organizations to develop, internally or externally, applications that tried to mirror their business processes, achieving an application that responded exactly to the company's processes, but with high development costs and high dependency to the entities or programmers who developed these custom applications. Applications developed internally by organizations are less flexible and more expensive to maintain and operate. With the year 2000 bug and the successive legal impositions that have been demanded year after year, companies have chosen to purchase ERP software externally.

One of the reasons is the excessive cost associated with the constant development of internally produced software. Another reason is based on the incompatibility or difficulty of these applications connected to other software. Even by imposing customers \ suppliers, who demanded the connection to their software, for example Electronic Data Interchange (EDI) and even because these applications are developed with technologies already outdated and with limitations.

There are authors who mention the term Enterprise System Software (ESS), such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) and Supply Chain Management (SCM) (Graeme Shanks, Peter B. Seddon, 2003), but in practice the companies that are currently developing ERP include all these applications as part of their ERP modules.

ERP systems have the following characteristics:

- Software with own development frameworks;
- Complex applications (Chou et al., 2005) and with modular design comprising the maximum of the company's processes, such as financial, logistics, production, etc.;
- Database management system (DBMS);
- An ERP is a generalist software that is quite customizable, flexible and expresses the best practices in the market;
- Systems that require large investments, both from the point of view of their licensing and in the consulting processes in the survey of the processes of the organizations, their implementation in the ERP framework and follow-up after implementation.

An ERP is a generic software that organizations should configure to their needs. ERP applications are rapidly replacing custom-built, in-house software. ERP systems have a significant impact on the organization of companies as well as on their strategy. Based on the structure that is based on ERP software, there is a direct and indirect standardization of the main processes, which allows a better communication experience between organizations. ERP systems and the Internet were probably the two most important information technologies that emerged in the 1990s (Graeme Shanks, Peter B. Seddon, 2003).

Researchers typically attribute the start of ERPs in the late 1960s. Companies felt the need to develop their own applications to control their inventory, resulting in Invent Control Systems (ICS).

Material Requirements Planning (MRP) followed. The term MRP was developed by IBM at the time (Robert Jacobs & "Ted" Weston, 2007), which were systems developed in the 1960s and 1970s by manufacturing companies, with the purpose of planning the necessary resources, specifically in production and production planning of manufactured products, management of raw material inventories and of the components necessary for the manufacture, as well as the management of the inventories of the final products.

Manufacturing Resources Planning (MRP II) emerged in the 1980s, with the evolution of MRPs, with the optimization of production processes, material planning requirements in production, and production planning itself, and already addressed the financial area.

In the early 1990s came a new concept of integrated software, Enterprise Resource Planning (ERP). The ERP allows companies to integrate the data of the entire organization (Davenport, 1998) and the fundamental of the ERP is to record the data of the transactions that are processed and registered in a database (Hawking & Sellitto, 2010).

Extended ERP (ERP II) comes about 2000 with the availability of ERP systems through Web services such as E-commerce and the integration of modules such as production, sales, design, logistics services, maintenance, Customer Relationship Management (CRM), accounting, Human Resources (HR) among others. Figure 3 shows the evolution of the ERP software.

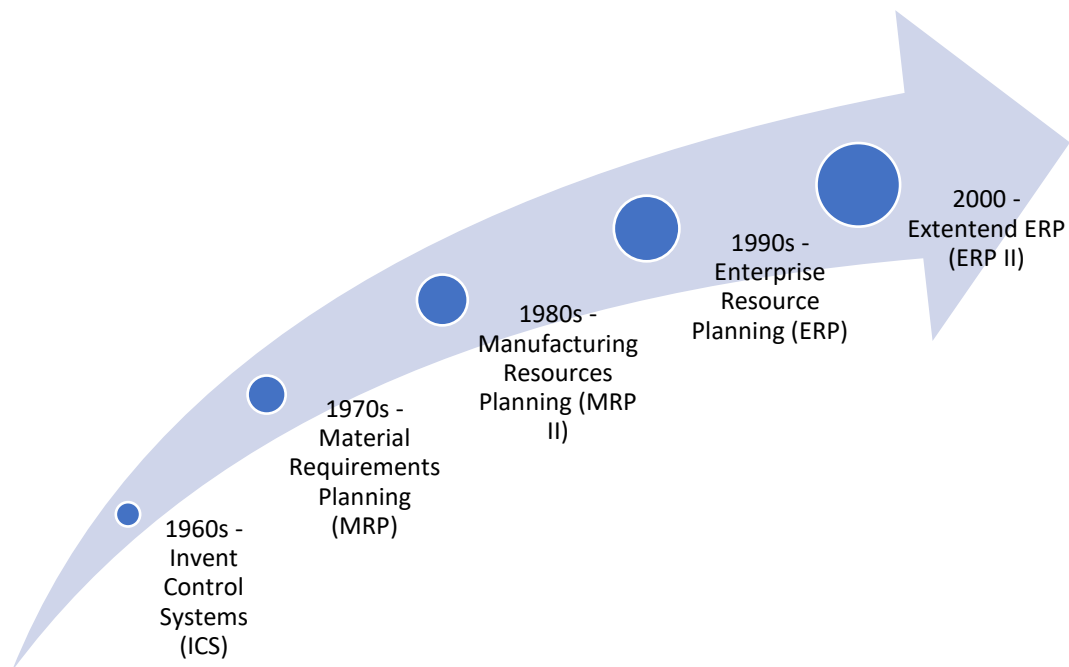


Figure 3 - ERP Evolution

2.2.1. Processes of the business areas

The ERP must record all data of the different phases of all processes of business areas of the organization. The need for automation of business processes is achieved in ERPs (Ganesh, K. , Sivakumar, 2014), especially using Business Process Management (BPM), an ERP module that, through operational workflows implemented in the applications, optimizes tasks and phases throughout each process. To achieve this, it is necessary to survey the processes, responsibilities and users that will be responsible during the phases of the company's processes. While workflows are essentially automation tools for the

different phases that documents should go through organizational processes, BPMs add greater integration with applications and tasks within the enterprise. The creation of flowcharts with the processes is essential for the implementation of BPM, with the decision maker's discrimination, responsible functions and the steps of the activities throughout the organizational processes of the companies. Figure 4 shows an example of a purchasing process. All processes of the organization must be registered in diagrams, to be clear to all employees.

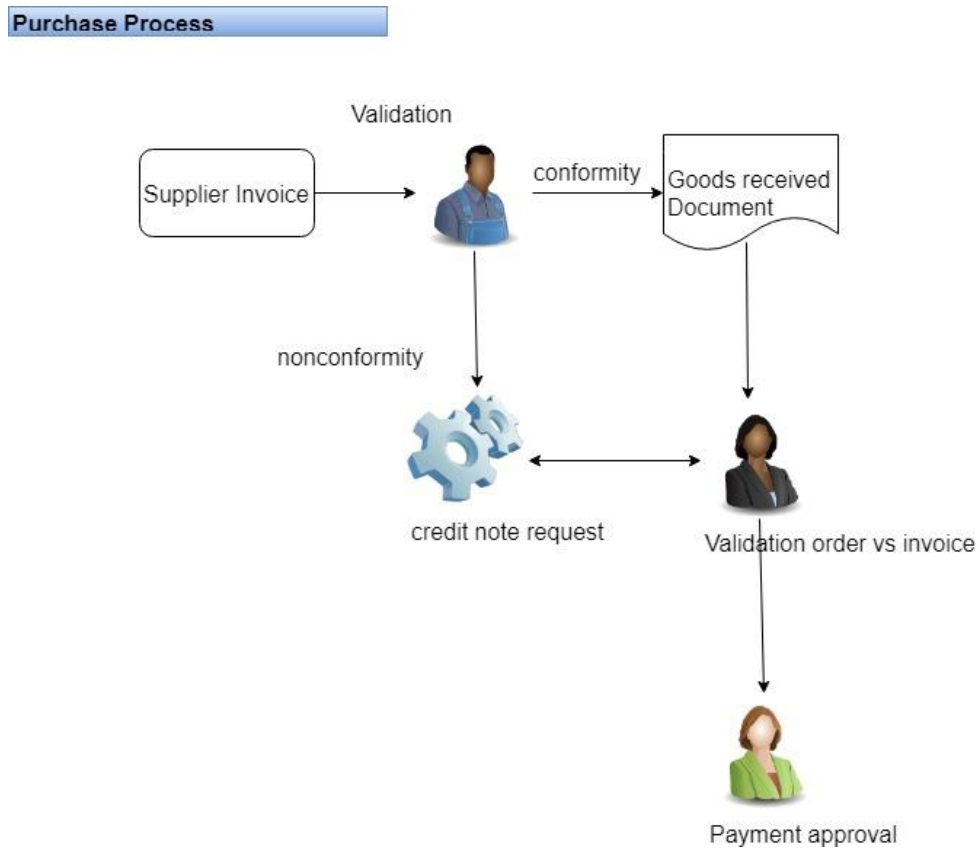


Figure 4 - Purchase Process

2.2.2. ERP Architectures and Technologies

Enterprise Resource Planning typically has a framework of its own. Figure 5 presents a case-by-case approach to the application of the ERP algorithm (Chou et al., 2005), using a variety of devices, such as computers or mobile devices, to data from a database of a database server.

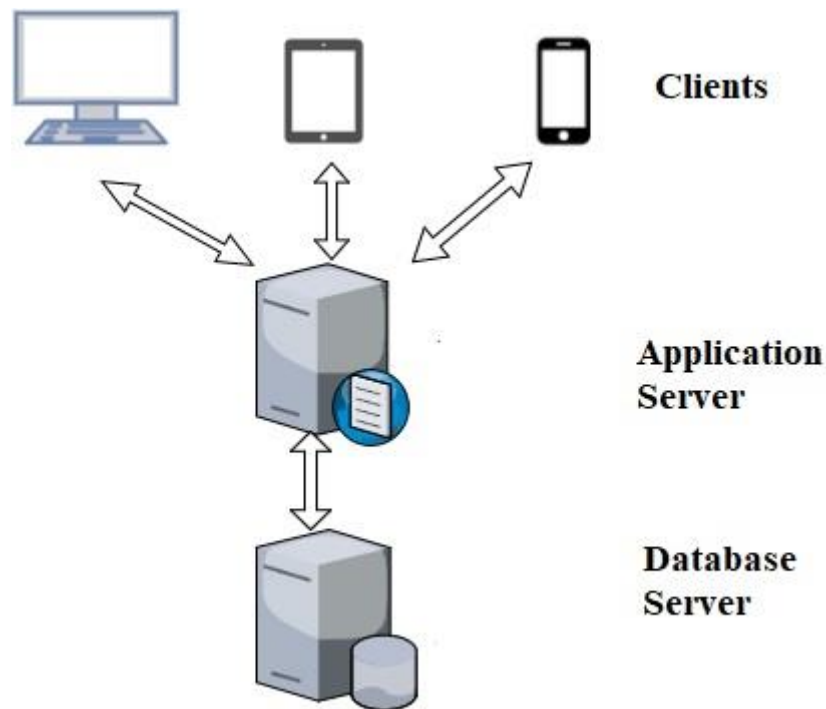


Figure 5 - ERP Architecture

ERP register transactions in a central and single database, with the advantage of only registering data once without redundancy or registering at different locations and in different databases. The most commonly used database engines are Microsoft SQL Server, IBM DB2, Oracle, and MySQL.

Mobility is now a reality and a necessity in organizations. ERP software developer companies are developing Web application accessing ERP data via Webservices and APIs in mobile applications or Web applications on any mobile device independent of the operating system. Considering the need to make the applications available in real time and in various locations and devices, the ERPs now have additional modules that make the software available through a web interface.

2.2.3. Advantages and disadvantages of an ERP

ERPs are integrated applications that enable you to enter data once in a process system and track it throughout the enterprise. That allows one to obtain information from customers, suppliers, production (Koupaei et al., 2016), purchasing, sales, human resources, accounting and management analysis.

An ERP system imposes its logic on company strategy, culture and organization (Davenport, 1998). What Davenport means is that ERPs are the result of the best

implementation practices over the years by companies that develop ERPs and thus attempt to standardize processes regardless of company organization and culture, as opposed to custom software. Also, (Graeme Shanks, Peter B. Seddon, 2003) stated that while ERPs are quite configurable, they can impose behaviour patterns on organizational processes.

ERPs now reflect the enormous experience gained over time in implementing the most varied processes in diverse types of organizations. Adapting ERPs to enterprises using parameter definitions is often called configuration (Davenport, Thomas H., Prusak, 1998). The customization of ERP by adding functionalities (usually through code development) to the software is called customization.

In the ERP customization there is, for example, the creation of new tables and new fields in the database, or the development of print reports or analysis of operational management, or creation of automatism in the processes. One way for organizations to avoid customization is to change their process to match the ERP. There is another risk here, the process changes are difficult, and the processes embedded in the software may not be appropriate for the organization's needs. The decision is in the customizations, configurations and process changes that will be best for the company.

ERPs are complex applications and their implementation requires large financial investments, time to implement and experience of use. The major implementation problems are the reconciliation of ERP technical requirements with the business needs of enterprises (Davenport, 1998). The costs of implementing an ERP are not only the costs of licensing the software, but also the cost of the services (consultant service hours) required for implementation, and the latter figure is usually quite high.

The cost of implementing an ERP can reach investment values, five to 10 times the cost of the application licensing involved (Davenport, Thomas H., Prusak, 1998).

The risk associated with the implementation of an ERP is great if there is not an involvement and commitment of the decision makers as well as the external team that is normally responsible for its implementation. The greatest risk is in surveying the company's processes to be implemented in the application. Experienced consultants should correctly interpret the processes and objectives to be implemented. ERP software is so complex that by even having an initial good implementation it is unlikely that there is a correct fit for the organization (Graeme Shanks, Peter B. Seddon, 2003). ERP software

not only has to mirror business processes, it also must fulfil legal obligations imposed by different countries.

2.2.4. Implementing an ERP

A problem in many organizations is that there are no well-defined organizational processes and are difficult to express through formal schematics either at high level or detail, i.e. knowledge of processes is being transmitted from collaborator to collaborator or from manager to manager and not are mirrored in formal registers.

The challenge of both the company and the ERP side is the survey of the company's business processes and which processes to transpose into the software. Organizations invest ERPs to access powerful information systems less costly than proprietary software development.

The biggest difficulty in implementing an ERP is the survey of company processes for software. Many companies have their businesses flowing according to history and in processes that are in the managers' minds, which are not documented. Often the processes are passed from people to people. Managers know how to proceed, but have difficulty translating what they know into paper. So, the consultants who do the survey are usually responsible for this translation. Process collection should be done across the enterprise, but it is the decision maker or decision makers involved who should validate.

Organizational processes often have to be changed to fit the ERP processes (Davenport, 1998). ERP are generic systems, standard processes implemented in most organizations. Software houses develop the structure of their software according to best practices, in their perspective and implementation experiences. In many cases the software allows greater efficiency, but in others this may not happen due to problems in its implementation. ERPs today are very flexible, allowing in their implementation a customization according to the company's processes, but there are always additional costs of this implementation, which often does not happen, and it will be the pre-established processes in the software that determine the new company processes. Many small companies take advantage of the ERP implementation to optimize their processes and leverage the knowledge and experience of the consultants in defining the new processes. If an implementation is planned someone must decide which applications (modules) will be implemented in each phase.

2.2.5. Critical Success Factors

The implementation of an ERP project is a case of project management and as such, much of the critical factors in its implementation have to do with project management. Funding for a project by the sponsor, good project management, communication throughout the company about what is being done and goals to be achieved, a medium-term perspective not only of the company's current processes but also processes that can be implemented in the future in the company and in the software. Thus, it is possible to know whether the software responds not only to the current requirements of the company but also to the additional processes that may be implemented in the future. The typical critical success factors of an ERP implementation project, second (Davenport, Thomas H., Prusak, 1998):

- Clear understanding of business needs;
- Top management support;
- Good project management;
- Experience, power and commitment of the responsible team (internal and external);
- Organizational preparation;
- Sufficient technical architecture.

2.2.6. Maturity and future of ERP systems

ERPs today have a very complex, comprehensive and deep development. Applications over the years have developed many features and respond well to most market needs.

The current challenge of ERP is user mobility. Business users today need to access the application not only in various locations but also access the ERP through various mobile devices such as tablets or mobile phones.

Through the implementations of webservices and APIs, ERP systems today have modules that respond to these needs, with mobile applications or Web applications.

2.3. Mobile

The use of mobile devices (smartphones and tablets) has had a generalized growth globally and Portugal is no exception, by analysing the graph in Figure 6, there is a great increase in recent years. as in the world in general Figure 7, where it is highlighted that the number of mobile users grows to the detriment of desktop PC users, either because of the mobility of people in their day to day, and due to the great technological development, that these equipment have undergone in recent years.

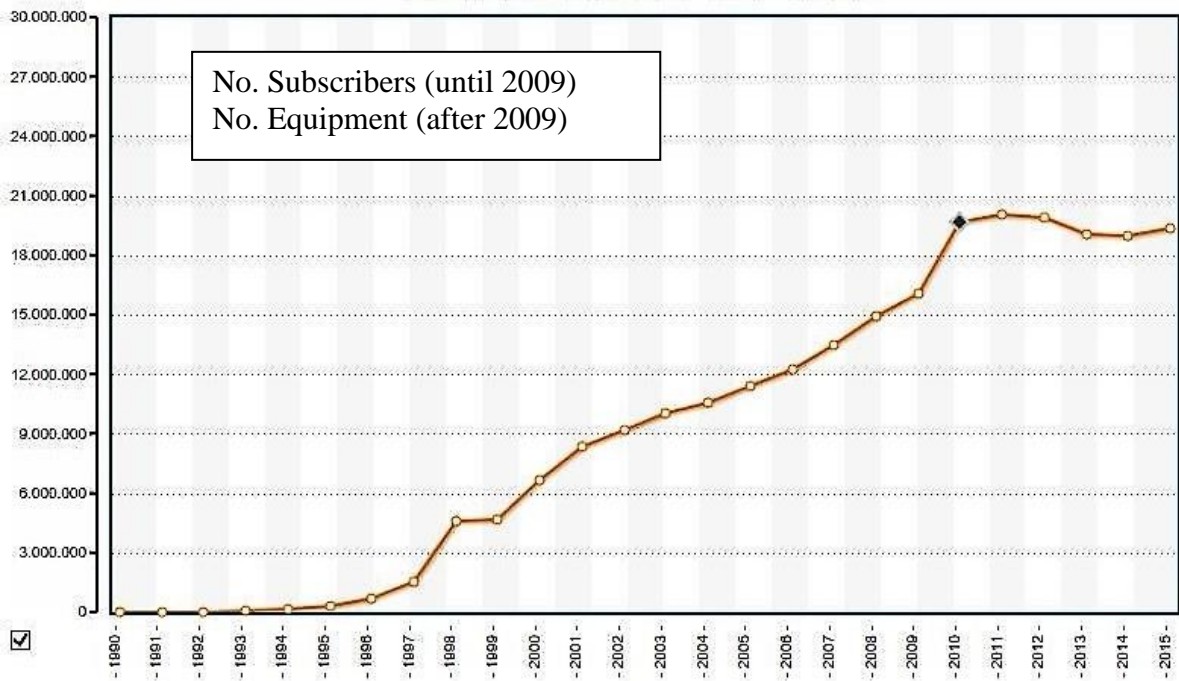


Figure 6 - N° Subscribers of mobile land service in Portugal (Portdata, 2018)

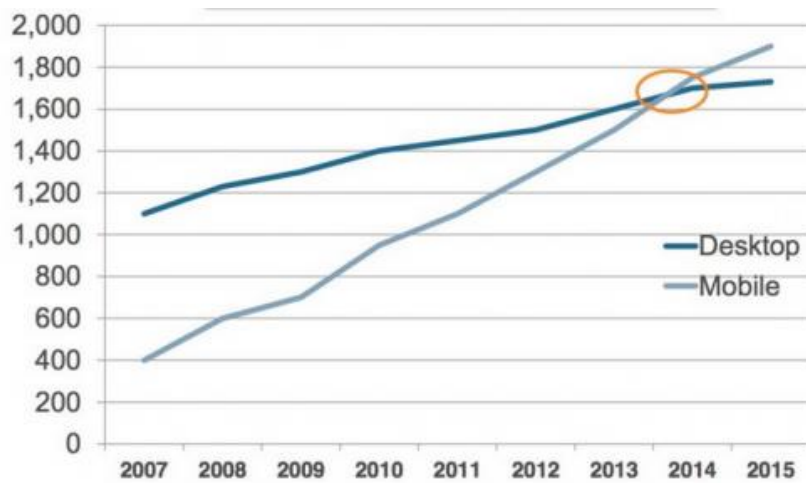


Figure 7 – Number of Global Computing Users (in millions) Source: comScore, Morgan Stanley 2012

The chart presented in Figure 8 shows how the computing market share evolved in the last years worldwide, where one can see that the market share for Desktop applications was initially far larger than the Mobile one, but decreasing until 2016 when there was a shift between the demand for Desktop and Mobile application development, following the growth on the number of devices owned by people of all socioeconomic levels.

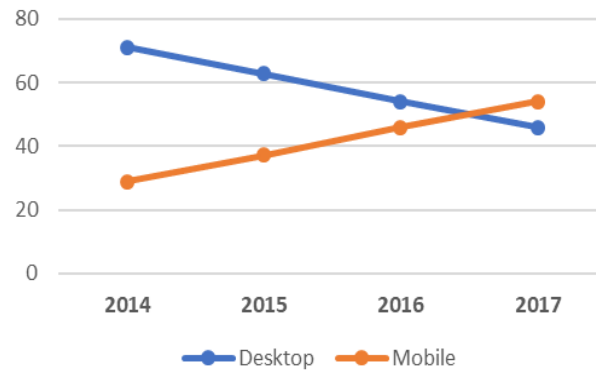


Figure 8 - Desktop vs Mobile market share since 2014(in %) (“Mobile Operating System Market Share Worldwide | StatCounter Global Stats,” 2018)

The growth of the mobile world globally has long surpassed the market share of desktop development. The technology innovation in mobile computing, along with the increased capabilities of the mobile devices, the dramatic improvement of the usability and look & feel of portable devices, together with their increasingly cheaper prices has resulted in a massive number of mobile devices in the market. The easy access to this technology and the rapid growth in the numbers of purchased mobile devices resulted in a high demand for mobile applications. The generalized growth of this type of devices was accompanied by the development of applications Figure 9.

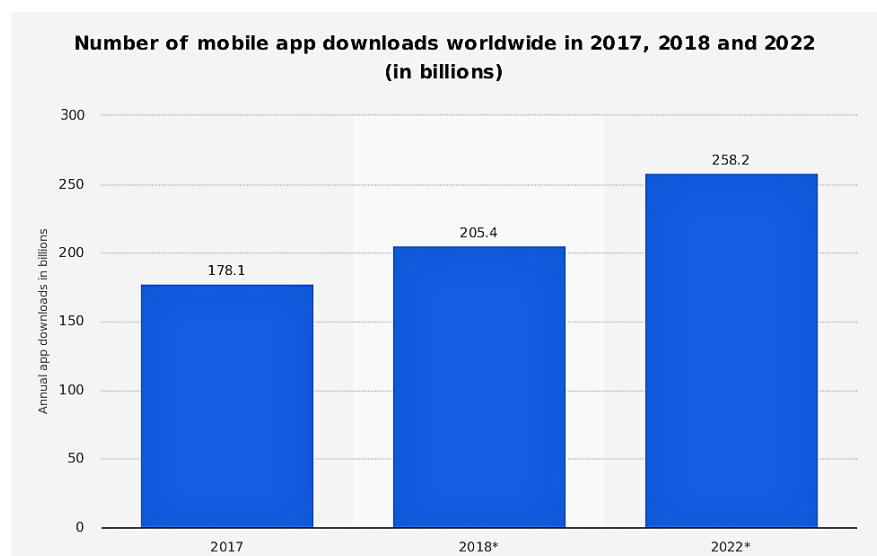


Figure 9 - Estimate on number of downloaded mobile apps Source: (Statista, 2018b)

The market for mobile devices is divided into two categories: hardware builders and software companies that develop operating systems and apps. Builders build, manufacture, assemble the various electronic components and configure an Operating System (OS) in their equipment to take full advantage of the various components such as the Wi-Fi camera, 4G, accelerometer, etc. of their devices. Companies that develop operating system software produce systems to meet the needs of the functionalities that hardware manufacturers and end-users seek (Perchat, Desertot, & Lecomte, 2013).

On the OS side there is the Apple iOS, Google's Android, Microsoft's Windows, Blackberry's RIM among others, where Android and iOS dominate the market Figure 10 according to (Statista, 2018b) and Figure 11 (“Mobile Operating System Market Share Worldwide | StatCounter Global Stats,” 2018). Each development platform uses different programming languages and interfaces, and, in the developments, it is necessary to have PC and MAC computers, as well as access to at least one mobile phone for each type of OS.

Thus, the development of applications for mobile devices is difficult due to the specifics of each platform and devices.

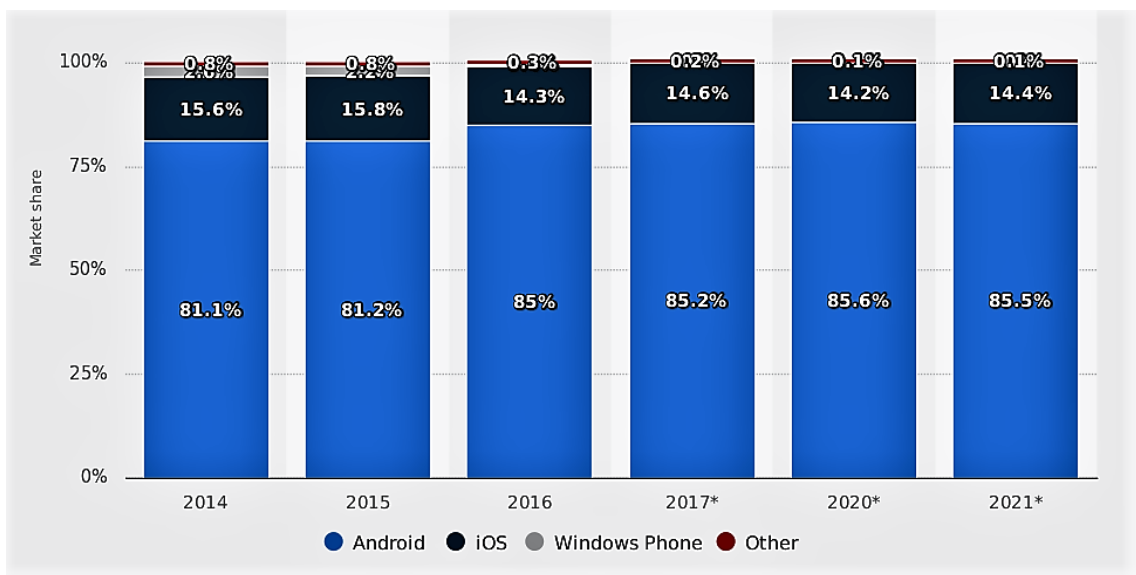


Figure 10 - Market share of operating systems in smartphones (in %) (Statista, 2018b)

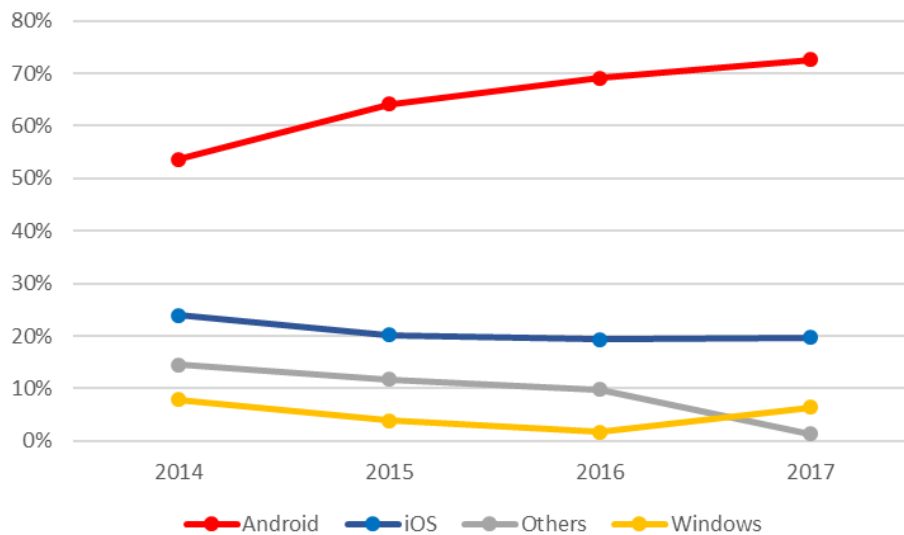


Figure 11 - Mobile Operating System Market Share (in %) (“Mobile Operating System Market Share Worldwide | StatCounter Global Stats,” 2018)

The proliferation of mobile devices, the constant advances in mobile computing, and the huge leaps in mobile Human-Computer Interaction (HCI) increased the desired demands to access ERP systems via mobile devices (mobile ERP) such as smartphones, tablet computers, and mobile handheld computers. Mobile ERP research is considered the latest research trend of ERP systems.

The usability of mobile ERP’s user interfaces (UIs) can be considered one of those research areas. Mobile computing has improved the way of doing today’s businesses. The ERP system is the core component of the mobile ERP. Mobile ERP system consists of three main components namely: the mobile application (mobile app) that access the back-end ERP system and was deployed on the mobile device; the mobile net communication; and the backend ERP system (Omar & Gómez, 2016).

2.4.Mobile Applications

Mobile application development is currently one of the most important skills a programmer can possess. Over the past decade there has been a burst of mobile devices; like smartphones, tablets and wearables, which gave rise to a full suite of mobile applications. In a time of mobile applications its creation is a difficult challenge for any programmer who needs to know various programming languages to develop for the devices. Languages such as Objective-C and Swift, for the development of iOS devices,

Java for Android devices and C# for Windows devices, among others. The emergence of Hybrid development platforms that use web development languages that take advantage of Native device functionality and are interpreted by any mobile device comes as an opportunity for the entire developer community.

The dynamics of the business market, whether because of globalization, the opening of markets, enormous competition or high technological development, mean that companies must have an efficient information system that can record operational data and transform them into information and knowledge to make business decisions.

Information systems technologies are a set of processes and tools (Bahrami, Arabzad, & Ghorbani, 2012) that help to achieve greater competitiveness in companies, risk perception and decision making. One of the characteristics of the information age is that there is excessive focus on mastering transaction data and not enough on turning it into information and knowledge that can lead to business results. Information systems in organizations register trillions of bytes of business transaction data to meet operational needs. Access to the information available in the ERP through mobile devices is a market need, considering the mobility of human resources.

The unprecedented growth and development of information and communication technologies (ICT) has influenced organizations. The business environment is increasingly complex, with functional units that require more and more data flow for decision making, timely and efficient procurement of goods, stock management, accounting, human resources and distribution of goods and services. In this context, the management of organizations needs efficient information systems to improve their competitiveness. Also, the high competition in the markets lead to the need to decide based on important information and quickly.

In today's dynamic and challenging business environment, access to ERP with mobile application solutions helps organizations in their operations and processes.

In ERP architectures, the most commonly used databases are Microsoft SQL Server, which is a scalable database engine that runs on Microsoft's Windows system. SQL Server is a database platform with enterprise-class data management with integrated business intelligence tools and provides the highest levels of security, reliability and scalability for applications. IBM DB2 is a database management system and provides a flexible and cost-effective platform to build robust solutions for enterprise applications. MySQL is

another widely used multithreaded, multiuser database engine such as SQL Database Management System (DBMS).

Enterprise expectations continue to increase around mobility. It's a cornerstone for many digital projects, journey to cloud and even analytics and digital marketing. Customers' mobile phones are a key data point for companies to track their interests, location, purchases etc, as well as offer user's real-time interactions. In the enterprise, the mobile is becoming the dominant computing device (Accenture, 2018).

2.5.Mobile Application Development Tools

The development of mobile applications led to the emergence of platforms and tools to take advantage of the Native features of the devices and the intrinsic particularity of their screens. The diversity of operating systems on mobile devices leads to the problem that programmers must grasp various programming languages and develop the same application for various systems and interfaces (Perchat et al., 2013), due to the incompatibility of systems, with the disadvantage of higher costs and longer development time. (Smutný, 2012) in 2012 categorized mobile applications into four different types; Native applications, Hybrid applications, dedicated web-specific applications for a specific platform and generic web mobile applications, which are mobile web pages that run on any platform.

Mobile devices have different screen sizes, resolutions and different aspect ratios, making application development difficult, (Holzinger & Slany, 2012) encourages programmers to clearly separate the User Interface (UI) definitions from the rest of the development code, in the sense of a cross-platform approach (El-Kassas, Abdullah, Yousef, & Wahba, 2017).

The development of mobile applications is a special case of software development because programmers must consider different aspects such as short development cycle, device capabilities, their specificities, such as screen size, UI design, security navigation and privacy (El-Kassas et al., 2017).

The development cycle consists of the following:

- Analysis of the idea or market requirement;
- The graphic interface design;

- The application development using tools and programming languages;
- Tests on different devices;
- The publication of the application in the devices or application store;
- Updates or new features are considered in new versions of the application.

2.6. The Native Approach

Native applications are developed using the specific operating system programming languages for which they are implemented. The development of mobile applications for various devices with different operating systems (OS) implies an additional cost in resources and greater difficulty in their development because there is a need for raising skills in different programming languages. There are many operating systems currently in the mobile device market, but the big players are the Android and iOS.

There are different SDKs for each platform and different tools and devices with different functionalities on each platform (El-Kassas et al., 2017), (Latif, Lakhri, Nfaoui, & Es-Sbai, 2016), as can be seen in Table 1. In fact, the only thing these operating devices have in common is that they all have a mobile browser that is programmatically accessible from the Native code (Charland & Leroux, 2011). The difficulty to develop mobile applications forcing the use of many different SDKs and frameworks motivate the implementation of cross-platform software development environments (Latif et al., 2016).

Table 1 - Native Development Environment per OS

	iOS	Android	Windows
Development Tool	X-code	Android Studio	Visual Studio
Programming Language	Objective-C/Swift	Java Code-Base	C#, C++

The Native code is usually compiled which makes it faster than languages like JavaScript that are interpreted. Native code is faster than JavaScript and HTML. The costs involved in this type of development for multi-platform are offset by better application performance and lower resource consumption (e.g., CPU, GPU, battery) by the devices that run, achieving a better end user experience.

Using Native SDKs, the programmer can access all the full features of the mobile device, without dealing with plugins and third-party dependencies. Native apps have

better user experience and performance but are not able to cross platforms (Que, Guo, & Zhu, 2017).

Native mobile applications are applications developed using the SDK and programming language specific to the mobile platform. The key limitation of these mobile applications is the inability to transfer applications to another platform, without writing the application from scratch (Bosnic, Papp, & Novak, 2017).

2.7.The Web Approach

The development of mobile web applications has arisen in the great development of technologies such as HTML5 and CSS using JavaScript language and leveraging the skills of programmers in web development. Web technologies are well suited to cross-platform application development because they are popular, standardized, relatively simple but powerful (Adinugroho, Reina, & Gautama, 2015).

The content of a web page is described by Hypertext Markup Language (HTML). HTML5 evolved from HTML and includes new attributes and behaviours. Apart from HTML5, the building blocks for most of the modern browser-based applications include JavaScript (JS) and Cascading Style Sheets (CSS) (Shahzad, 2017).

For mobile web development several sets of tools, platforms and libraries can be used.

2.7.1. jQuery mobile

jQuery mobile (“jQuery Mobile,” 2018) is a jQuery foundation project that is one of the most used open source JavaScript libraries in web development. JavaScript is a most popular programming language in the world (Bera, Mine, & Lopes, 2015).

With the development of JavaScript, CSS and HTML5 technologies, web sites have become more responsive to adapt to mobile devices and their different resolutions and sizes. jQuery Mobile is fast, small and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript (“jQuery Mobile,” 2018).

In modern web development AJAX, a short name for Asynchronous JavaScript and XML, is a natural mixture of several technologies and is commonly applied for

asynchronous communications with web server (Paulson, 2005). jQuery is a fast, library that simplifies the development of dynamic HTML web pages and uses AJAX capable of exchanging data with a server, and update parts of a web page, without reloading the whole page. jQuery is very light-weight, easy to use and flexible as compared to other JavaScript frameworks (Wajid, Junjun, Akbar, & Mughal, 2018) . jQuery Mobile is a user interface (UI) framework geared to mobile applications that is built on jQuery and is a cross-platform to design to multiple devices. jQuery Mobile is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices (He, Zhang, & Fang, 2017), (Latif et al., 2016).

2.7.2. Bootstrap

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Providing a responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery (Shahzad, Sheltami, Shakshuki, & Shaikh, 2016).

Also, the evolution of the browsers in the last years in the sense of better use of these technologies, have increased the potential of the development of web applications. The web approach is based on web browsers for mobile devices (Latif et al., 2016). These web applications are accessed on mobile devices through the device browser. The emergence of open source libraries of responsive front-end development, i.e., the graphical interface adapts to the different screen sizes in the different devices contributed to the expansion of this type of applications. An example is the Bootstrap platform with its responsive grid system through component libraries. The Bootstrap grid system creates a responsive layout that fits to every type of device, as can be seen in Figure 12.



Figure 12 - Bootstrap Responsive Web Layout

Bootstrap includes CSS and JavaScript code that provide baseline style and layout rules for common web page elements such as grids, navigation bars, buttons, and dialog boxes (Walker & Chapra, 2014). Bootstrap also incorporates the normalize CSS style

sheet to eliminate many common cross browser compatibility issues associated with webpage styling and formatting rules. Currently Bootstrap is in its version 4 with a major rewrite of almost the entire project, with some big disruptive changes in relation to version 3. The Bootstrap grid system creates a responsive layout that fits and adapts to every type of device.

2.7.3. Angular Framework

Angular is an advanced front-end framework released by the team at <https://angular.io/> (“Angular - One framework. Mobile & desktop,” 2018). Is a client-side technology, written entirely in JavaScript. JavaScript is a fundamental piece of modern Web applications. Angular works with the long-established technologies of the web (HTML, CSS, and JavaScript) to make the development of web apps easier and faster than ever before. Angular is a popular JavaScript MVC-based framework to construct single-page web applications (Ramos, Valente, & Terra, 2017). Angular is a platform that makes it easy to build applications in the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop (<https://angular.io/>). It enables you to build a rich front-end experience, quickly and easily. The browser fetches the web pages and display them to the user browser, gets the HTML text of the page, parses it into a structure that is internally meaningful to the browser, lays out the content of the page, and styles the content before displaying it to you (Bott, 2014).

With Angular, one is not just building the structure, but also constructing the interaction between the user and our app as a web application. Angular takes care of advanced features that users have become accustomed to in modern web applications, such as:

- Separation of application logic, data models, and views;
- Ajax services;
- Dependency injection.

It also augments HTML to give it Native Model-View-Controller (MVC) capabilities. MVC is a software architecture pattern that separates representation from user interaction. This choice, as it turns out, makes building impressive and expressive client-side applications quick and enjoyable.

The Angular source code is made freely available on GitHub under the MIT license.

Instead of merging data into a template and replacing a DOM element, Angular creates live templates as a view. Individual components of the views are dynamically interpolated live. Generally, the model consists of application data and functions that interact with it, while the view presents this data to the user.

In the modern web applications architecture like AngularJS, the software developer moves its focus from traditional programming patterns and structures to focus on the actual business logic and user interface design, as seen in Figure 13.

Angular was built by a team of engineers at Google, is a JavaScript-based open-source front-end web application framework, adding new features and syntax, that compiles to plain JavaScript and run on all browsers. Angular it is a TypeScript-based front-end web application platform that makes it easy to build applications with the web. TypeScript is fundamental for the development on Ionic and others Hybrid platforms like Meteor. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

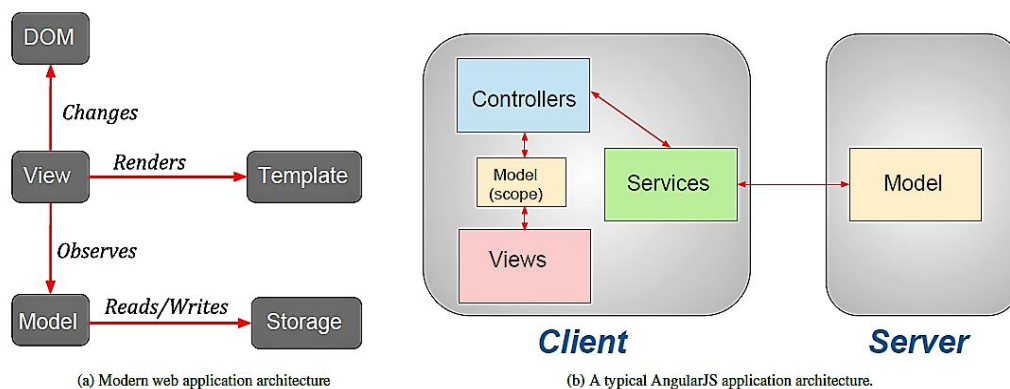


Figure 13 - Application architecture for a modern web application Source: (Shahzad, 2017)

Modern web applications architecture like Angular, the software developer stay focuses on the actual business logic and user interface design.

In the sense of greater usability and faster web applications, a new concept of Single-Page Application (SPA) emerged. SPAs are web-based applications which load a single HTML page and dynamically update the page content as the user interacts with the

application through menus and side bars (Shahzad, 2017). These applications offer a more-Native-app-like experience to the user.

These applications have a lower development cost than Native applications, but they have a limitation because they do not have access to the Native features of the devices, such as the phone, the camera and the contacts. Another limitation is that they cannot be marketed in distribution stores, such as the Apple App Store or the Google Play Store.

2.8. The Hybrid Approach

Although it's possible to develop a Native app for each platform individually (such as developing with Objective-C/Swift for iOS and/or Java for Android) and to deliver a great UX, when targeting several platforms, the costs of such approach can be prohibitive, both in terms of time to market (TTM) and total cost of ownership (TCO) across the app's lifetime. To help control and lower these costs, different cross-platform development technologies have evolved to produce platform-specific app packages from a shared code base (Torre & Calvert, 2016).

For the development of mobile applications for the broad spectrum of mobile devices on the market programmers had the need to learn various programming languages like Objective-C and Swift for iOS devices, Java for Android devices and C # for Windows devices, among many other languages and devices. These development languages were designed for applications taking advantage of Native features such as GPS, camera, phone, etc. There is thus a need to rewrite the entire application for devices with different operating systems. The disadvantage is undoubtedly the increased time, the associated cost and the difficulty in finding resources with experience in the different platforms of development. Developing applications that can be used across platforms is extremely difficult. All development platforms have different software development kits (SDK) with different programming languages, as can be seen in Table 2.

Table 2 - Programming languages vs platforms

Platform	Programming Languages
iOS	Objective-C, Swift
Android	Java
Windows Phone	C#, C++

With the development of HTML5, JavaScript and Apache Cordova, and the need for a solution that allows a shared language to develop applications for a wide range of devices, Hybrid applications emerged, combining web technologies HTML, CSS and JavaScript language accessing Native capabilities of the devices.

Hybrid apps use web technologies such as HTML5, CSS3, JavaScript language and Cordova plugins to access API platform of the device, to provide a Native wrapper for containing the web-based code, and a generic JavaScript API to bridge all the service requests from the web-based code to the corresponding platform API. It is the Native wrapper that enables Hybrid mobile apps to be packaged, deployed, and distributed across platform (Malavolta, 2016), (Wargo, 2012), (Huynh & Truong, 2017). Hybrid mobile app development require frameworks, like Apache Cordova, Ionic among others to create cross platforms Hybrid apps with Native looks and feel.

Hybrid application development platforms are increasingly popular in device-independent application development (Kudo, Yamauchi, & Austin, 2017). Single-Page Applications (SPAs) are web-based applications which load a single HTML page and dynamically update the page content as the user interacts with the application through menus and side bars (Shahzad, 2017). Hybrid apps can run anywhere the web runs - on a desktop (Windows, Linux, Mac OS, or other) or mobile browser, as a mobile app, or Progressive Web Apps (PWA).

2.9.Mobile App Comparison Table: Native vs Web vs Hybrid

Native, mobile Web, or Hybrid? Which one to choose? The right choice depends on variety of factors, like;

- Which skills do you have in the team?
- What features, and Native functionalities do you need?
- When do you need it?
- How much is your investment?
- Which platforms and stores will be your target?

Native application provides best performance and user experience, but Native applications are very expensive, in most cases a business will not need a Native app. If

you don't need Native features of the devices a web approach could be the cheapest way. If the target will be a multiple platform then Hybrid app could be a good possibility. On Table 3 shows a comparison table that will help the reader to understand.

Table 3 - Comparison Native vs web vs Hybrid

	Native	Web	Hybrid
Speed	Since they are developing through Native SDK and specific language system, they are faster. Take less CPU and GPU, of the device	Since they need to be compiled, take more time, slowest	Since they need to be compiled, take more time than Native
Functionalities	All, full functionality of mobile device	Only Web browser, does not access device features like phone, gps, accelerometer, etc.	Almost all functionalities of the device through Cordova plug-in.
Time cycle	More time to develop than web or Hybrid. Need to develop in several languages and SDK to achieve different platforms Expensive in development, more time consuming, development more 2 times for other platforms	Faster to deploy	Faster to deploy than Native and developing to all platforms
Team work	Need to have team with skills for each SDK platforms	Web skills are easier, easier to get developers	Need to have team with skills for Hybrid platforms
Difficulty	Difficulty to get developers of each SDK	easier	Easier than Native
Platforms	Only one platform for each SDK	all	all
Cost	More expensive, More time consuming, developers costlier	Cheapest, Less time and cheaper developers	Cheaper than Native
Stores	Deploy in each platform store	no	Yes, can be deploy to all platforms stores

2.10. Mobile Hybrid development

The high demand for creating mobile Hybrid applications, caused the emergence of several development platforms over time.

2.10.1 Adobe PhoneGap

PhoneGap is an open source platform for creating cross-platform Native applications. PhoneGap started in 2008 at iPhone DevCamp by the company Nitobi who started the project to simplify multiplatform development (Wargo, 2012). The PhoneGap development platform (<https://phonegap.com/>) can reconcile web development and its associated technologies like HTML5, CSS and JavaScript with access to the Native functionality of mobile devices (Charland & Leroux, 2011). The PhoneGap platform contains code for interacting with the underlying operating system and passing information back to the JavaScript application that runs on the web view, such as geolocation, accelerometer, and more. Apache PhoneGap is the commercial version of Apache Cordova.

2.10.2 Apache Cordova

Apache Cordova, is an open-source mobile development framework, created by Nitobi, and purchased by Adobe in 2011. The PhoneGap code was contributed to the Apache Software Foundation (ASF) under the name Apache Cordova (“PhoneGap,” 2018). So, Apache Cordova is the open source version of PhoneGap. Apache Cordova enables software developers to create applications for mobile devices using HTML5, CSS and JavaScript (Huynh & Truong, 2017). The UI of a Cordova Application is a WebView which occupies the complete screen of the device, and it will run in the Native container. The WebView will remain the same in all operating systems, only the Native container will change according to the mobile platform (Novac, Marczin, & NOVAC, 2016), (Torre & Calvert, 2016). Hybrid applications can access the mobile device resources through JavaScript using a bridge that communicates between the JavaScript code and the source code of the device (Kudo et al., 2017). Cordova accesses the resources of devices from different mobile platforms through a JavaScript plug-in. Cordova, there is no compile process. Apache Cordova supports a set of default plugins called core plugins. These plugins allow us to access device capabilities such as the battery, camera, contacts, storage, etc. (Bosnic et al., 2017).

2.10.3 Xamarin Platform

Xamarin Platform was founded in May 2011, by the engineers who developed Mono, Mono for Android, and MonoTouch which are cross-platform implementations of the Common Language Infrastructure and Common Language Specifications. Common Language Infrastructure is an open specification developed by Microsoft that describes executable code and a runtime environment (Novac et al., 2016). Xamarin was purchased by Microsoft on February 2016.

Developers can use client-side technologies to build client apps themselves, using specific frameworks and patterns for a cross-platform approach. With Microsoft technologies, developers can build Native (Native-single-platform using languages like Objective-C and Java with Microsoft Azure SDKs, Native and cross-platform apps using Xamarin, .NET and C#), Hybrid (using Cordova and its variants, see Figure 14, or websites (ASP.NET), depending upon their decision factors (Torre & Calvert, 2016).

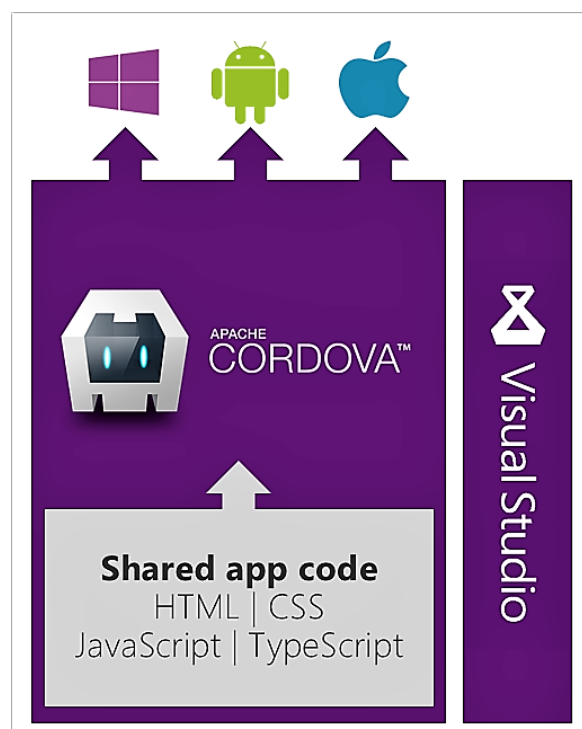


Figure 14 - Visual Studio tools for Apache Cordova

Using HTML, CSS, and JavaScript, developers can take advantage of the skills they developed while building websites and apps to create mobile apps for iOS, Android, and Windows with Apache Cordova. Most developers achieve nearly 100 percent code reuse while using the Cordova shared JavaScript API to access Native device options such as cameras, calendars, and other hardware capabilities. A Cordova app is composed of the

same HTML/JavaScript/TypeScript code that you can compile for each platform (iOS, Android, and Windows) (Torre & Calvert, 2016).

Xamarin is a cross-platform mobile application development platform, which uses C# language in development and .NET. With Xamarin is possible to develop mobile applications in C#, as shown in Figure 15 for iOS, Android and Windows.

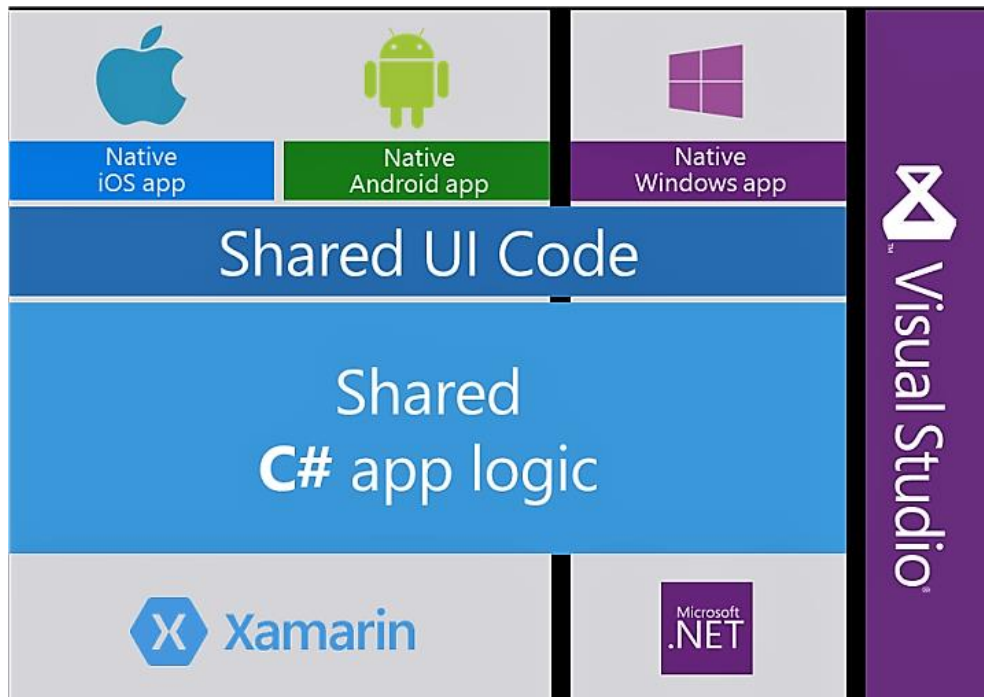


Figure 15 - C# cross-platform with Xamarin and .Net [Source: (Xamarin, 2018)]

According to (Xamarin, 2018), it is possible to use 75% of code development on average on all mobile development platforms. It is currently the Microsoft platform for the development of cross-platform applications. For cross-platform development with Xamarin one can use several IDE like Visual Studio on PCs or Visual Studio Code and Xamarin Studio on a Mac.

2.10.4 Firebase Mobile Platform

As a client-side framework, Angular alone is not enough to build a full back-end web app. It's often difficult to know when to sync our data with the back end and how to handle the changes and potential conflicts of data between versions of modified content.

Let us imagine two instances of our application running at the same time. What if both instances are trying to edit the same data? Without handling this case, one can get into trouble, especially when building the front end for a complex web application.

Using Firebase, it is easily possible to add a back end to our Angular App. Featured on the Angular.js home page, Firebase is quickly becoming the standard for Angular persistence.

Firebase is a real-time back end for building collaborative, modern applications. Instead of requiring one to focus on building custom request-response models with a server-side component where it is possible to manually worry about data synchronization, Firebase allows one to get the app up and running in minutes.

It is possible to build a data-backed web app entirely in Angular that can scale out of the box and update all clients in real time. Data that is stored in Firebase is standard schema-less JSON, which makes it incredibly easy to save data models of any type into Firebase. If a device loses network connection, Firebase continues to allow access to locally cached data and seamlessly synchronizes changes with the cloud when the device comes back online.

The Firebase client libraries and REST API provide easy access to that data from any platform. Although focusing specifically on Angular, this fact means that Native apps or other server-side apps can reach the data that Angular has saved. By default, the firebase service returns a simple JavaScript object.

Firebase platform is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. Firebase is part of Google Cloud platform solutions (“Firebase,” 2018). Modern tools require special consideration for the challenges developers face on mobile: server less capabilities, a cloud-first data model capable of persisting data even when the device is offline, low-latency access to media anywhere in the world, and real-time data synchronization across all mobile platforms (“Mobile Solutions | Google Cloud Platform,” 2018). Angular and Firebase are complementary tools for server less development for web and mobile apps.

2.10.5 React

ReactJS is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words, it enables web developers to write mobile applications that look and feel truly “Native,” all from the comfort of a familiar JavaScript library (Eisenman, 2015). React is a JavaScript framework that allows us to develop Native mobile applications for

both iOS and Android, without having to write code in either Swift or Java. React is a declarative, efficient, and flexible JavaScript library for building user interfaces (“React - A JavaScript library for building user interfaces,” 2018). React provides an excellent developer experience over normal mobile applications. This is mostly because of the great set of developer tools that comes built in with the framework. Since it's just JavaScript, all those tools will feel familiar to the web developer.

2.10.6 Meteor

Meteor is a full-stack JavaScript platform for developing modern web and mobile applications (“Build Apps with JavaScript | Meteor,” 2018). Meteor is an open-source technology that relies on the Model-View View-Model (MVVM) software design pattern to address the stateless (volatile data), unidirectional (communication overhead), and unreliability (network bottlenecks) concerns of the Internet. The framework uses JavaScript on the client, server, and data modelling sides, enabling developers to quickly produce more features in a shorter amount of time by collaborating on a single programming language (Adams, Persaud, Acworth, Adams, & Hamadeh, 2013). MongoDB as its primary database, which is a popular NoSQL (Not Only Structured Query Language) that is not dependent upon relationships between entities, and instead focuses on managing unstructured data.

Meteor, a rapid prototyping framework for JavaScript, is one of the go to frameworks for creating a cross-platform app that works on iOS, Android, and the web.

2.10.7 Ionic Platform

The Ionic platform was founded in 2012 by Drifty Co, to facilitate the development of mobile applications for web developers. The development framework allows developers to use web technologies to create Hybrid mobile applications (“Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular,” 2018), (Apps, Justin, & Jude, 2017). The Ionic platform is today one of the most popular mobile development technology platforms in the world with over 32,990 stars on GitHub a source code hosting platform. The Ionic framework is an open source Framework built with HTML, CSS and JavaScript for the development of Hybrid mobile applications. It is the most popular and the top pick by many developers (Huynh & Truong, 2017). Apps can be built with these web technologies and then distributed through Native app stores to be installed on devices by leveraging Cordova.

jQuery mobile and Bootstrap were by 2013 two of the most popular tools at the time for web development and mobile applications (Griffith, 2017). With the evolution in performance and new browsers functionality in mobile devices, they have brought advantages to the development of Native-like apps for web developers in mobile development.

At the same time, AngularJS appeared as an open source JavaScript platform and a TypeScript-based front-end by the Google team that further enhanced web development. AngularJS was very popular with programmers and seemed to provide a seamless response to reusable JavaScript and HTML 5 components for the web.

JavaScript was originally created by Brendan Ike at Netscape and then was adopted by Microsoft as Jscript. With different versions of a language for different browsers, there were soon an urgent need to standardize the language. The European Computer Manufacturer Association (ECMA), is the governing body that provides the ECMAScript specifications for JavaScript browser implementations. ECMAScript 1 specification was release in 1997, followed by ECMA 2, ECMA 3, ECMA 4 and ECAM 5in 2009. In 2015 ECMA 6 came out.

ECMAScript is a scripting language standard and specification, JavaScript, Jscript and ActionScript are languages that are based on ECMAScript standard. ES6 or ES2015 is the most recent version of ECMAScript / JavaScript and have major updates since ES5 in 2009. ES6 has backward compatibility, has a modern syntax and new features. Some browsers required transpilers to compile ES6 to ES5. New features like, let and const declarations (to declare variables), classes, template string, promises, arrow functions among others. With the new ECMAScript 6 companies that make web browsers had new guidelines for use JavaScript. The latest versions of browsers like Chrome, Firefox and IE are compatible with the most part of ES6 (<https://kangax.github.io/compat-table/es6/>), but if one wishes to assure that our app runs in all browsers our ES6 code needs to be transpiled in to ES5 before. Transpiled is the process of taking ES6 code and converting it into ES5, so it can be read by browsers.

TypeScript programming language is a superset of JavaScript, that is an extension of JavaScript Figure 16, adding new features and syntax, that compiles to plain JavaScript and run on all browsers. In 2015 JavaScript evolves from ES5 to ES6 the new generation of JavaScript with new features for object-oriented development.

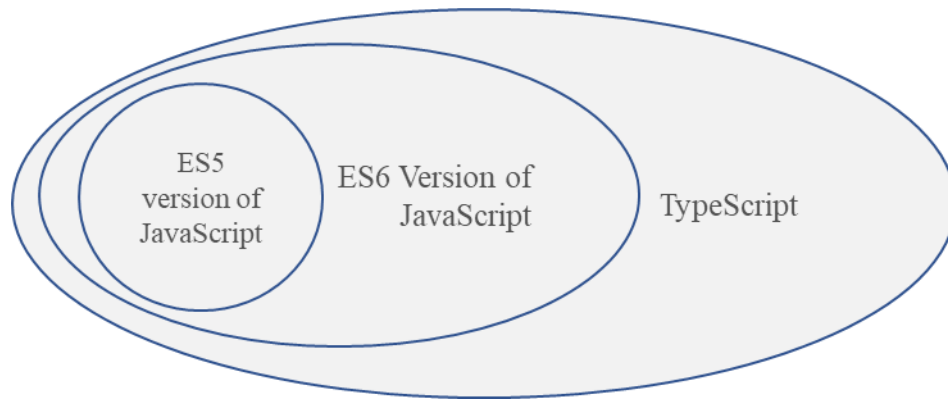


Figure 16 - TypeScript is a superset of JavaScript

There was a need to create transpilers to translate the new JavaScript syntax for ES5 syntax for the browsers to understand. Transpilers, or source-to-source compilers, are tools that read source code written in one programming language and produce the equivalent code in another language with the evolution of JavaScript, the Angular team saw an opportunity to create a structure that suits the future development of the web as mobile with the Angular 2 taking advantage of the new ES6. As expected the Ionic team emerged with the Ionic 2 platform taking advantage of the Angular 2 in 2015.

Angular 2 was rebuilt with the goal of running better on mobile devices. Therefore, Ionic 2 applications are faster and can handle more complexity than Ionic 1 apps (Griffith, 2017).

The year 2017 was a year of great change and development, for the Ionic platform and for the technologies that are associated with it. AngularJS went from version 2 to Angular and then to version 4 and Ionic accompanied its platform to the Angular version 4 with the output of the version Ionic version 3, all these evolutions aimed at the best performance of the mobile applications created. Ionic uses the Angular in its development and platform structure and is essential for the development of the application.

The Ionic framework is the platform for building Hybrid mobile applications for iOS, Android, Windows Phone, and other mobile platforms. Hybrid applications are built for cross-platform use, using web tools. This means you can develop the application once and deploy it to multiple platforms without having to change the code base. Benefits of this are a shorter development time and less complexity. Hybrid apps run within a browser-based web view, which impacts their performance. Ionic has been optimized to the point where a speed difference is very small. Another downside is that Ionic requires

plug-ins to access some Native components and the components you may need may not yet exist. Ionic has near Native performance, one code base and near Native functionality.

To develop with Ionic Framework, one should understand and be familiar using HTML5, CSS3, Sass, Angular and JavaScript. It is necessary to understand the concept object-oriented programming (OOP) concept, such as classes, inheritance. It is possible create a project in any editor like Microsoft Visual Studio Core, Atom, Sublime and others. Ionic has a command line interface (CLI).

In the development in Ionic on debugging, Google Chrome browser, as it provides a great development environment using the developer tools. web browsers; the most common browsers today include Chrome, Safari, Mozilla Firefox, and Internet Explorer. At their core, they all basically do the same thing: fetch web pages and display them to the user, the browser gets the HTML text of the page, parses it into a structure that is internally meaningful to the browser, lays out the content of the page, and styles the content before displaying it to you (Bott, 2014). On Figure 17, the Hybrid applications architecture and usage of Apache Cordova tools is displayed.



Figure 17 - Hybrid applications architecture and usage of Apache Cordova tools (Bosniac et al., 2017)

Ionic uses the Angular and Cordova, in its development and platform structure and they are essential for the development of the application. Built on Apache Cordova and Angular, Ionic brings the features and functionality of those frameworks while providing a generous library of pre-built components and device plugins to choose from (Griffiths, 2016). Cordova acts as a container for running a web application written in HTML, CSS, and JavaScript. Typically, Web applications cannot use the Native device functionality like Camera, GPS, Accelerometer, Contacts, etc. With Cordova, developers can access the Native device functionality and can package the web application in the device installer format (Huynh & Truong, 2017).

With Hybrid framework like Ionic, apps can run on any platform or device from a single codebase. They can also run in a regular browser as a Progressive Web App (PWA), what is a major advantage over Native development.

The Ionic Framework is one cross-platform development framework with more than five million developers in over 200 countries, with 4 million apps built.

The development of mobile app with Ionic has dependency of plugins like Apache Cordova if it is needing to access features of mobile device. Cordova project is a layer that sits in-between Ionic and the mobile operating system. It wraps up our JavaScript, HTML, and CSS and provides API-specific bindings for the operating system. This is really the communication layer between our application and the mobile OS beneath it.

Chapter 3 – Application Concept and Application Development

3.1. Development of the mobile application CodeGT

An ERP software is currently a set of essential strategic tools that have a direct impact on the success of any organization, from collecting, registering and accessing information to decision making within the organization. Considering employee mobility, access to organization's information systems from mobile devices inside and outside the premises is fundamental, for operational, legal and analytical reasons.

The purpose of the mobile application CodeGT is to allow the access by mobile devices to the company ERP, in particular the elements of the Transport Documents. The application will not require the printing of these documents, which is of great interest to achieve a dematerialized management. This is complemented by additional features that will be provided by the developed mobile application, namely delivery registration (digital signature of the recipient), access to local (by geolocation) and the consultation of the elements (customer card, materials and equipment involved). This app will be developed in an enterprise context for use in real customers.

The interest in organizations in this development in understanding these processes from several points of view:

- For business is a market need, considering not only the legal impositions, but also as access to information from a mobile device.
- It will allow to take advantage of the Native functionalities of the mobile devices, namely the phone, the global positioning system (GPS) and the camera between others.
- Considering Portuguese legal obligations, about Transport Documents in the sense of accessibility of information registered in the ERP.

An ERP software is a complex application, with lots of information and difficult to interpret in the user interface. Mobile application designers should evaluate how users perform each task and remove non-essential functionality that hinders the user experience (Newhook, Jaramillo, Temple, & Duke, 2015).

The application developed has the purpose of meeting the creation of a market need to obtain specific information regarding the consultation of legal elements required in the transportation of goods in Portugal. So, this application will only be an add-on that will

make this information available to the user by limiting the amount of information needed to present on the mobile device to the location of the delivery of the goods, such as the company name of destination, address, postal code, Portuguese Government Tax Authority (AT) code. As extra features there are the GPS location of the recipient's address and the drive-to function, from where one can find a final address destination. The application also records the delivery receipt with the signature of the recipient as proof of delivery made.

3.1.1. Transport Documents

The shipment and transport of any cargo in Portugal has a set of mandatory legal obligations that are required to be followed. Namely, any transport requires being accompanied by a document that clearly describes the cargo contents, its source and destination among other information. Additionally, this information needs also to be reported to the Portuguese Tax Authority (AT) for taxation purposes. The AT institution then acknowledges having received this information by issuing a unique code, named “AT code”, which is required to be included in the Transport Document.

Documents are corporate processes that are registered in the ERP. The Transport Document is a document that must accompany the movement of goods in Portugal, in operations carried out by taxable persons, or companies. The Transport Documents are processed by taxable companies who are the consignors of the goods and by the holders of the goods before the start of transport. All taxable persons with a turnover exceeding EUR 100 000\year are obliged to communicate the Transport Documents by electronic data transmission or on AT web site or by telephone to AT. It is always necessary to communicate the Transport Document with a certified software or by telephone before starting the transport. According to Article 4 of the Outstanding Goods Regime, in Portugal, the Transport Document must contain:

- Corporate name, address and tax identification number of the sender of the goods;
- name, firm or company name, address and tax identification number of purchaser of the goods;
- commercial description of the goods and quantities;
- loading and unloading sites;
- the date and time when the transport begins.

The information regarding Transport Documents are registered in the ERP software which, through webservices, communicates with the Portuguese Tax Authority (AT), which provides the application a validation code, designated by “AT code” (legal obligation in Portugal). Any transport must always be accompanied by this “AT code” for validation by the authorities. An example of a Portuguese Transport Document can be seen in Figure 18.

SIGESCOM Guia de Transporte N 12
 Guia de Transporte Série II / GT 2010M912 ORIGINAL

Sigescom - Soluções Integradas de Informática, Gestão e Tel.
 Rua Poeta Bocage, 20-7E
 1600-581 LISBOA Lisboa
 Contribuinte N.º: 506852792
 Conserv. Registo Comercial: Lisboa
 Capital Social: 5.000,00

SIGESCOM, LDA
 Edifício Bocage, Rua Poeta Bocage, N.º20-7E
 Lisboa
 1600-581 LISBOA
 N.º Contribuinte: 506852792

Data : 2010-10-29 Data Entrega dos bens: 21.10.2010 Página 1 de 1

Software PHC - Processado por programa certificado nº 0006/AT (20180611)-Este documento não serve de fatura

Referência	Designação	Qtd.
TV	TV 55" 4K	1,0

Modo de Expedição : Código AT: 987654321
 Local de Carga: Nova Moura Local de descarga: SIGESCOM, LDA
 Data de carga: 21.10.2010 Hora de carga: 10:00 Hora de descarga: _____

Figure 18 - Example of a Transport Document with AT code from ERP PHC

3.1.2. Legal Transport Document Requirement

The fulfilment of the legal obligations for the communication of the elements in the Transport Documents, of goods in circulation in Portugal by the companies, obliges that the organizations register and communicate in advance the materials in circulation. This legal obligation began in 2012 with Ordinance No. 363/2010 of June 23, which has, however, undergone several changes by Decrees No. 22-A / 2012, dated January 24, and with Ordinance No. 198/2012 (Autoridade Tributaria, 2012).

3.2. Development Requirements

For the development of the mobile application, some requirements were needed. On the server side a computer with Microsoft Windows server is required for its development

and hosting web server IIS. A database server on Microsoft SQL Server, and an Application Server with the ERP software installed. The list of requirements;

- Microsoft Windows Server 2016 Standard, with IIS (Internet Information Services) web server for the API.
- Microsoft SQL Server 2016, database server that provides ERP database, on Windows server machine. The ERP PHC software only works with Microsoft SQL database.
- ERP PHC (PHC, 2018), ERP software system.
- Node.js (<https://nodejs.org/en/>) (“NodeJS,” 2018).
- Ionic (<https://ionicframework.com/>) (Ionic, 2018), Hybrid platform used on development of mobile app.
- Apache Cordova (<https://cordova.apache.org/>) (“Apache Cordova,” 2018).
- Firebase (<https://firebase.google.com/>) (“Firebase,” 2018).
- Visual Studio (<https://www.visualstudio.com/>)
(<https://visualstudio.microsoft.com/vs/>) (Microsoft, 2018b).
- Visual Studio Code (<https://code.visualstudio.com/>) (Microsoft, 2018a).
- Java JDK and Android Studio for create and build Android PacKage (apk). APK is the package file format used by the Android operating system for distribution and installation of mobile apps.
- A MAC computer and Xcode software, for create and build iOS App Store Package (ipa). IPA file is an iOS application archive file which stores an iOS app.
- DevApp Ionic mobile app for testing the app in a mobile device without installing it.

Development of mobile application steps;

- A RESTful Web API development in C# on Visual Studio 2017 was created.
- Mobile application development CodeGT on Ionic platform.
- Mobile app Ionic DevApp app for testing.

3.2.1. Architecture of Mobile Application

The Architecture of the Hybrid mobile application CodeGT is on Figure 19, that shows the Microsoft SQL Database that registers all data from ERP. The webservice was created to support the communication to the mobile app, and the authentication through the Firebase API OAuth validation of the user to access information. Firebase Cloud Storage is used to store all signatures of receivers of the goods, and Firebase Hosting to host the application.

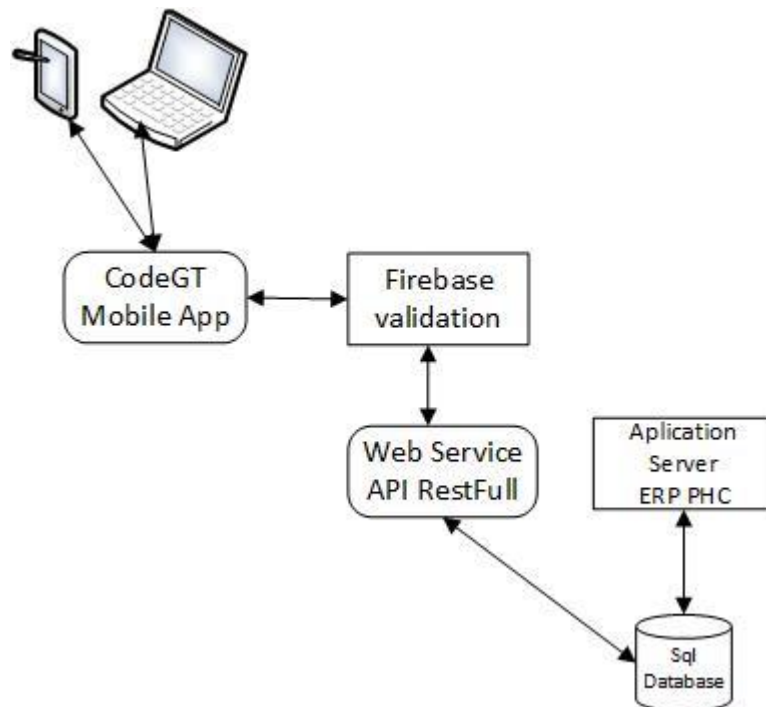


Figure 19 - CodeGT Architecture

3.2.2. Database

The development uses a Microsoft SQL relational database on a Windows database server. For the implementation, some dummy data was created. The starting point was a demonstration database. Some data was created for testing and implementing the mobile application. The ERP software has hundreds of tables on database but only two tables were used from that database. One to get the records from all the customers (customer table) and the other one to get information from the documents (Transport Documents) table.

The PHC Software has a complex database structure (with hundreds of tables and thousands of fields) on Figure 20. There, one can see only the UML from the two tables and only the fields that were object of this application on the table cl (customers) and ft table (documents). The relationship between tables is one to many, i.e. one customer to several documents.

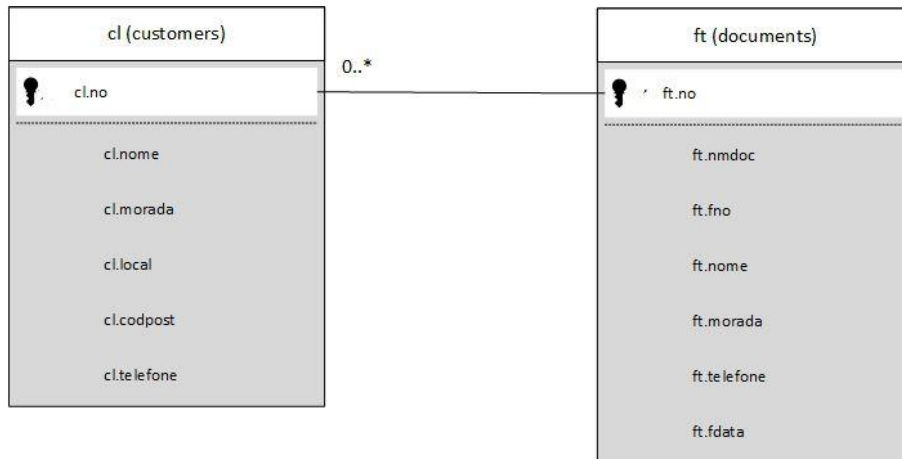


Figure 20 - UML from table cl and ft

3.2.3. Server Side

On server side was created a Visual Studio project ASP.NET MVC Web API in C# language. There was a server-side web service running and providing RESTful API to clients (mobile devices), in the various mobile devices through a Hybrid application developed on the Ionic platform. The term API stands for Application Programming Interface (API), the ASP.NET Web API is a framework that allow to build HTTP web services on the ASP.NET framework. On Figure 21 one can see the MVC pattern.

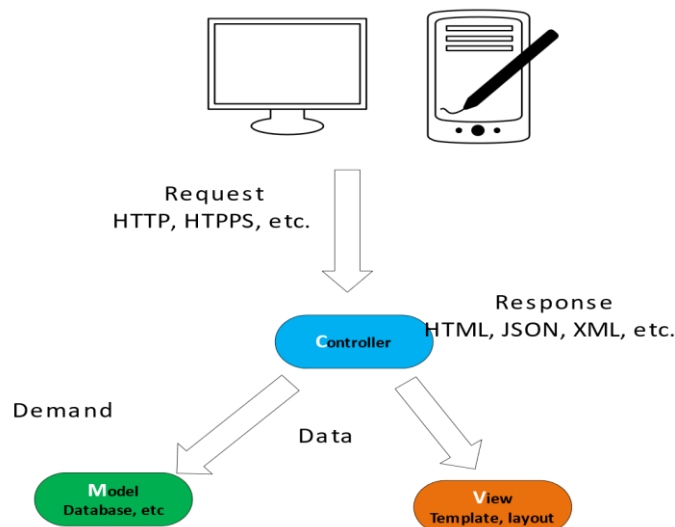


Figure 21 - The MVC pattern

For this project Representational State Transfer REST (REST) services were created with the SQL (Structured Query Language), Entity Framework, MVC, ASP.NET web API technologies. These services can be consumed by a wide variety of clients such as: browsers; mobile applications; desktop applications and Internet of things (IOTs); devices or objects that have an IP address and can communicate over the Internet with other devices and objects. REST is an architectural pattern for creating services. REST architectural pattern specifies a set of constraints that a system should adhere to.

REST constrains:

- Client Server, client sends a request and the server sends a response;
- Stateless, the communication between the client and the server must be stateless, which means that the server should not retain customer related information. The client request must contain all the information necessary for the server to respond to the request;
- Cacheable, requests that are repeated relative to the same information can be cached for better performance and faster response;
- Uniform Interface, defines the required interface between the server and the client;
- Layered System;
- Code on Demand(optional).

Windows Communication Foundation (WCF) is also a form of web services, in which it uses Simple Object Access Control (SOAP) but its structure is more complex than the REST web service.

Restful APIs use http methods, and can return JSON, and can model Create, Read, Update and Delete (CRUD) operations. But that's not specifically what makes an API restful. What REST is, is all about modelling your API around resources. And allowing clients to perform operations on those resources. In the context of REST, a resource is any object in your APIs design domain. In many cases, a resource will correlate exactly with a row in a table or an object in the data base. But that's not always the case. So instead of having API endpoints that are verbs, or represent actions you can take in the system, the endpoints in a restful API represent resources or collections of those

resources. HTTP Methods are the verbs that act on resources in a RESTful API. The HTTP methods for RESTful are GET, PUT, POST and DELETE.

Security is a major concern for any API or application today, and it's crucial to consider security early in the development of your project. It's much easier to design security from the beginning rather than trying to bolt it on at the end. There are two major types of security that need to be considered for an API built on asp.net core. The first is transport security, which means keeping the connection between the client and the server secure. Secondly, application security, which covers things like authentication and making sure users are authorized to perform actions in your system. The first and most important is HTTPS. By enabling HTTPS or SSL support, clients will be able to connect to the API with an encrypted connection. Once it's turned on, the next step is to force all clients to use HTTPS by redirecting any HTTP requests to HTTPS. The biggest component of transport security is enabling HTTPS, also called SSL or TLS. On Project properties enable SSL support. If a client accesses our API over regular HTTP, they're redirected automatically to HTTPS. On controllers, you can use the `require HTTPS` attribute. For example, I can open the root controller and add `require HTTPS`. However, this only enforces HTTPS for this one controller. It is possible to apply the attribute to the entire application at once by adding it as a filter in the start-up class. In the startup class, within the `configure services` method and the `AddMvc` section, it is advisable to require HTTPS for all controllers and add this to the filters collection.

Secondly, add some additional security headers to all API responses, such as the HTTP strict transport security or HSTS header. This will guaranty even greater security for clients that support those headers.

For the development of the API, was installed Microsoft Visual Studio Professional 2017, downloaded in <https://visualstudio.microsoft.com>.

In the development in Visual Studio (Microsoft, 2018b) was used the MVC technology, C# language, to make available a web service API RESTFUL, in format JavaScript Object Notation (JSON) to the requests of the mobile devices querying the database information. After creating the Data Model and Controllers, Figure 22 shows part of the developed code. Only the Transport Documents were selected, i.e., documents of type 2 and 33.

```

27 public IEnumerable<ft> Get(int ndoc= 0)
28
29 {
30     using (ErpDBEntities entities = new ErpDBEntities())
31     {
32         // to return only last 2 days
33         var d1 = System.DateTime.Today.AddDays(-300);
34         switch (ndoc)
35         {
36             case 2:
37                 // only Guia de Remessa
38                 return entities.fts.Where(e => e.ndoc == 2 && e.fdata >= d1 ).ToList();
39             case 33:
40                 // only Guia de Transporte
41                 return entities.fts.Where(e => e.ndoc == 33 && e.fdata >= d1).ToList();
42             default:
43                 return entities.fts.Where(e => (e.ndoc == 33 || e.ndoc==2) && e.fdata >= d1).ToList();
44         }
45     }
46 }
47
48

```

Figure 22 - part of Code for GuiasController in C#

3.2.4. Application mockup

The mobile application starts with a splash screen, then comes up a login screen for user validation. After that the app opens with a list of customers, where the user can select and see important information regarding each customer, inclusive the address, phone number and the location on google maps.

On the next page on Figure 23, one can see a User Interface (UI) mockup of the mobile application.

On the hamburger menu it is possible to choose other functionalities like the Transport Documents available to delivery. Here, after selecting it becomes possible to see information regarding the document and then to see the driving plan from specific initial location to the destination of delivery. Following that, and on the screen then becomes possible to register the signature of the receiver of the goods.

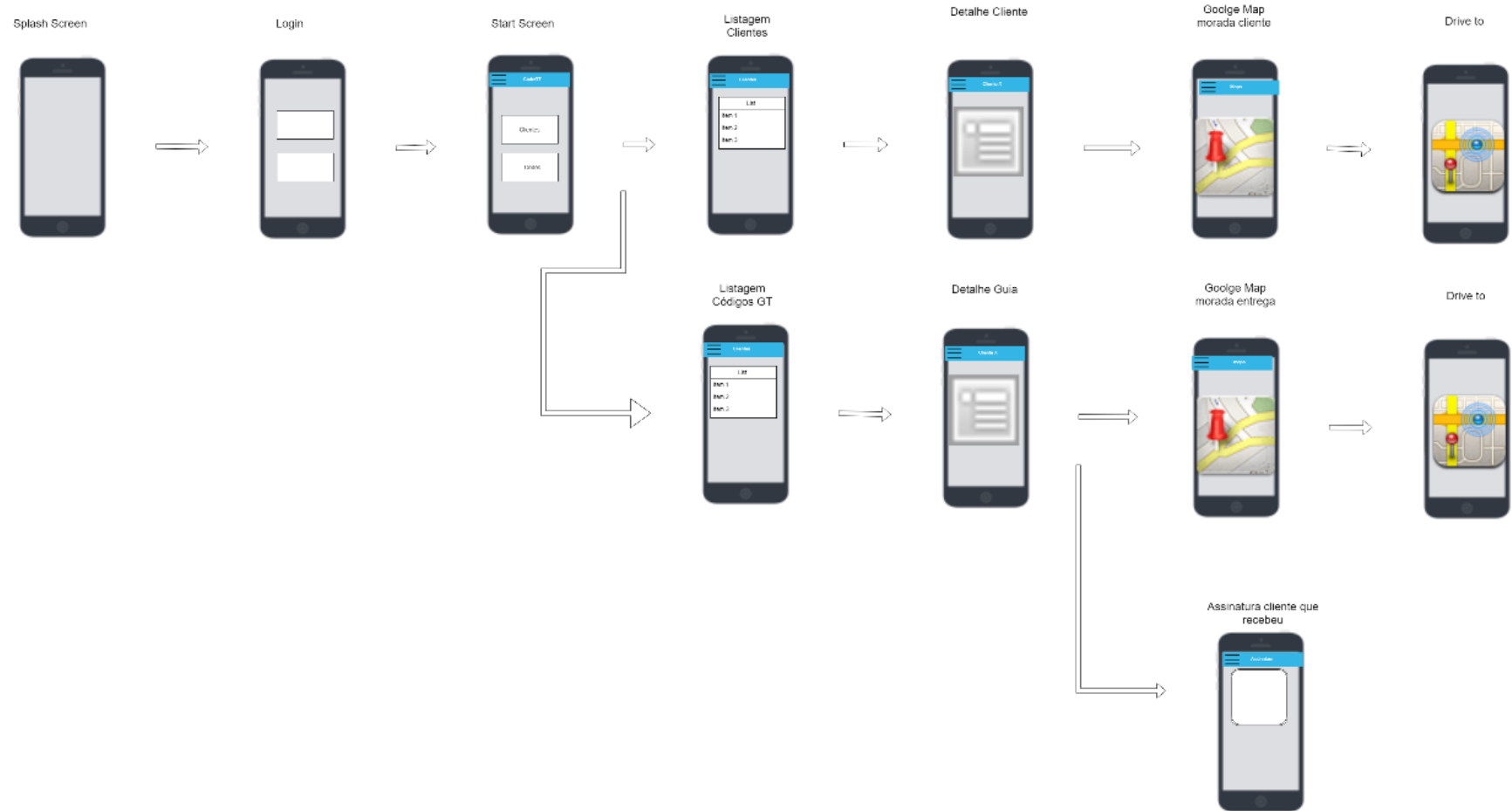


Figure 23 - Application Mockup

3.3. Mobile development

The development of the CodeGT mobile application requires a Windows, Mac or Linux computer. In this case a computer with a Windows 10 Pro operating system and a Mac with High Sierra macOS system were used, most recent systems in 2018. A Mac was needed it for building the iOS app.

The first requirement was to install Node.js through the executable downloaded from <https://nodejs.org/en/> (“NodeJS,” 2018) version 8.9.4 page for the respective operating system Figure 24.

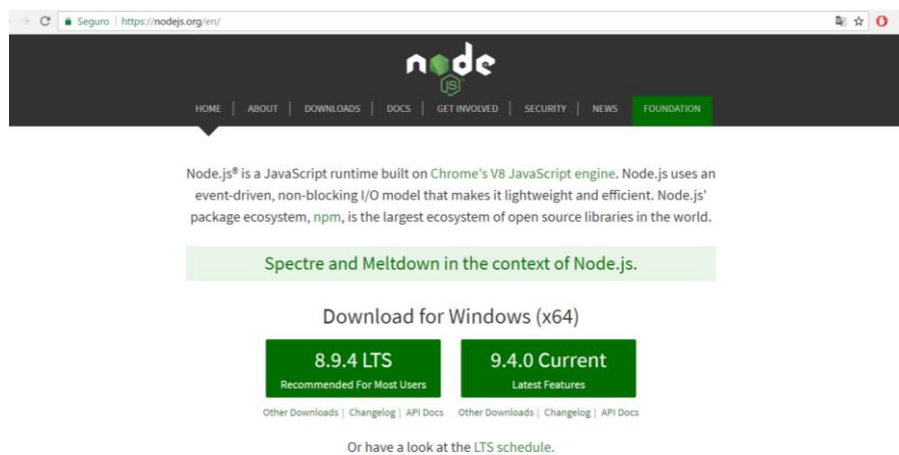


Figure 24 - Node.js site

Then through the command line interface (CLI) on Figure 25, the Ionic v3.19.1 and the Cordova v.8.0.0 were installed through the command;

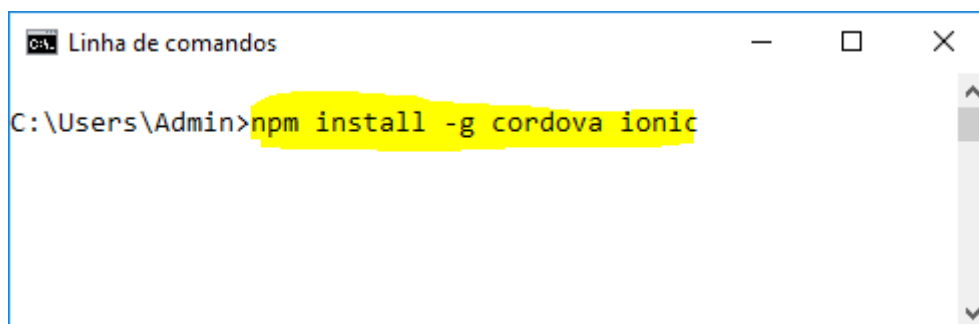
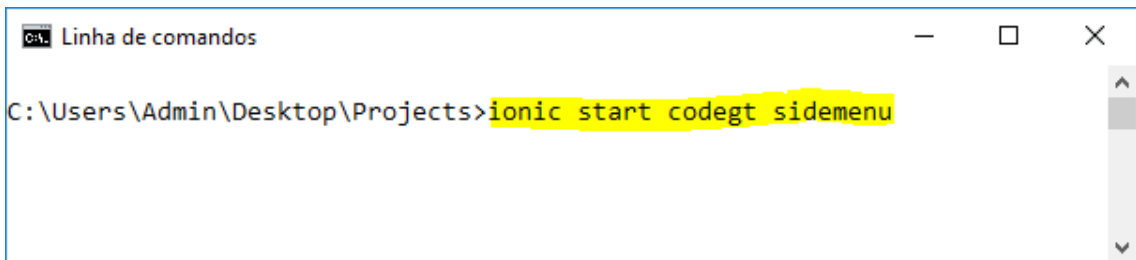


Figure 25 - Ionic CLI

Ionic provides several templates, to start:

- tabs: a three-tab layout.
- side menu: a swipeable menu on the side.
- blank: a bare starter with a single page.

To create the Ionic CodeGT app, a side menu was chosen Figure 26;

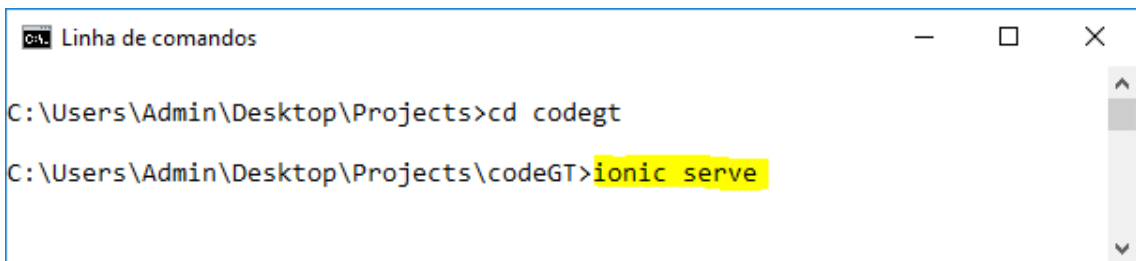


```

C:\Users\Admin\Desktop\Projects>ionic start codegt sidemenu
  
```

Figure 26 - CLI command to create CodeGT app

To run the app, it is necessary to change to the directory create, CodeGT and run ionic serve command to start the app Figure 27 and view app in the browser Figure 28. A web server runs locally and a very interesting functionality from Ionic platform is live code, meaning that by changing the code automatically it becomes possible to view the result on the browser immediately.



```

C:\Users\Admin\Desktop\Projects>cd codegt
C:\Users\Admin\Desktop\Projects\codeGT>ionic serve
  
```

Figure 27 - Ionic CLI command to start app

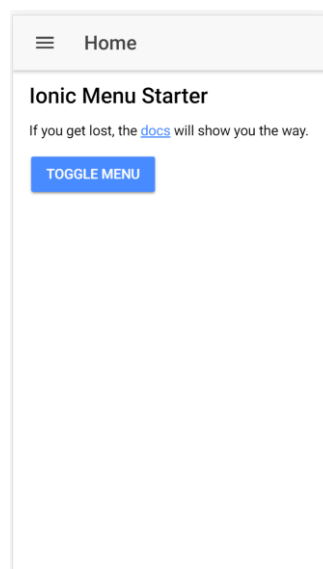


Figure 28 - Ionic side menu app created

The Ionic platform creates automatically a file structure for us Figure 29. The file structure is similar to Angular.

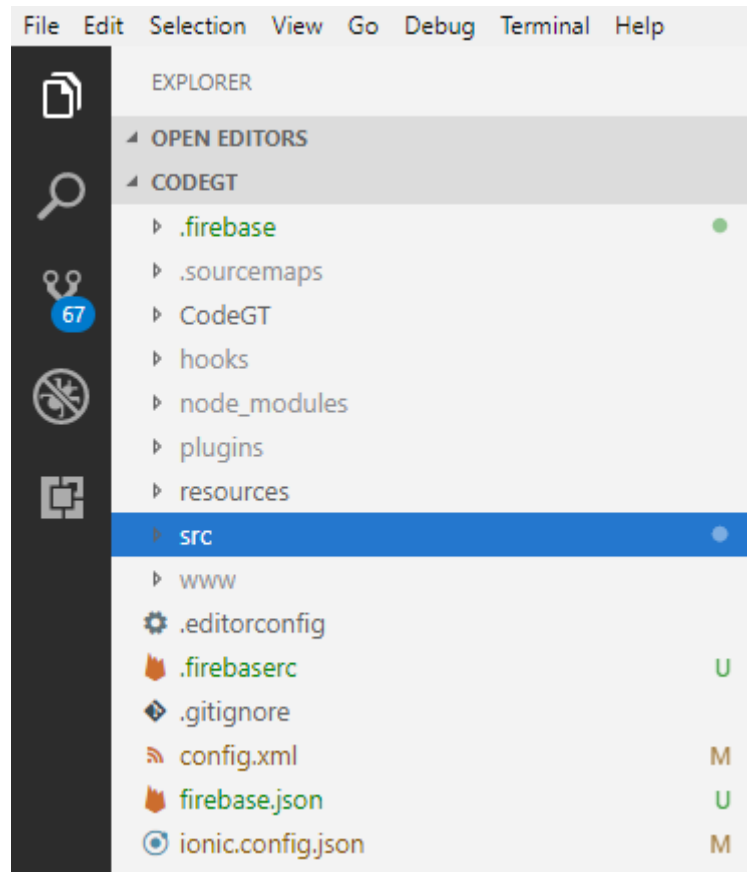


Figure 29 - Ionic, files structure

The source code editor used for the development of the Hybrid mobile application was Microsoft Visual Studio Code downloaded at (Microsoft, 2018a) <https://code.visualstudio.com/>.

The `src` directory is where most of the development takes place Figure 30. The `src/app/app.module.ts` is the entry point of our app. The root component of our app has control essential of the rest of the application, in `src/app/app.component.ts`. This is the first component that loads in our app.

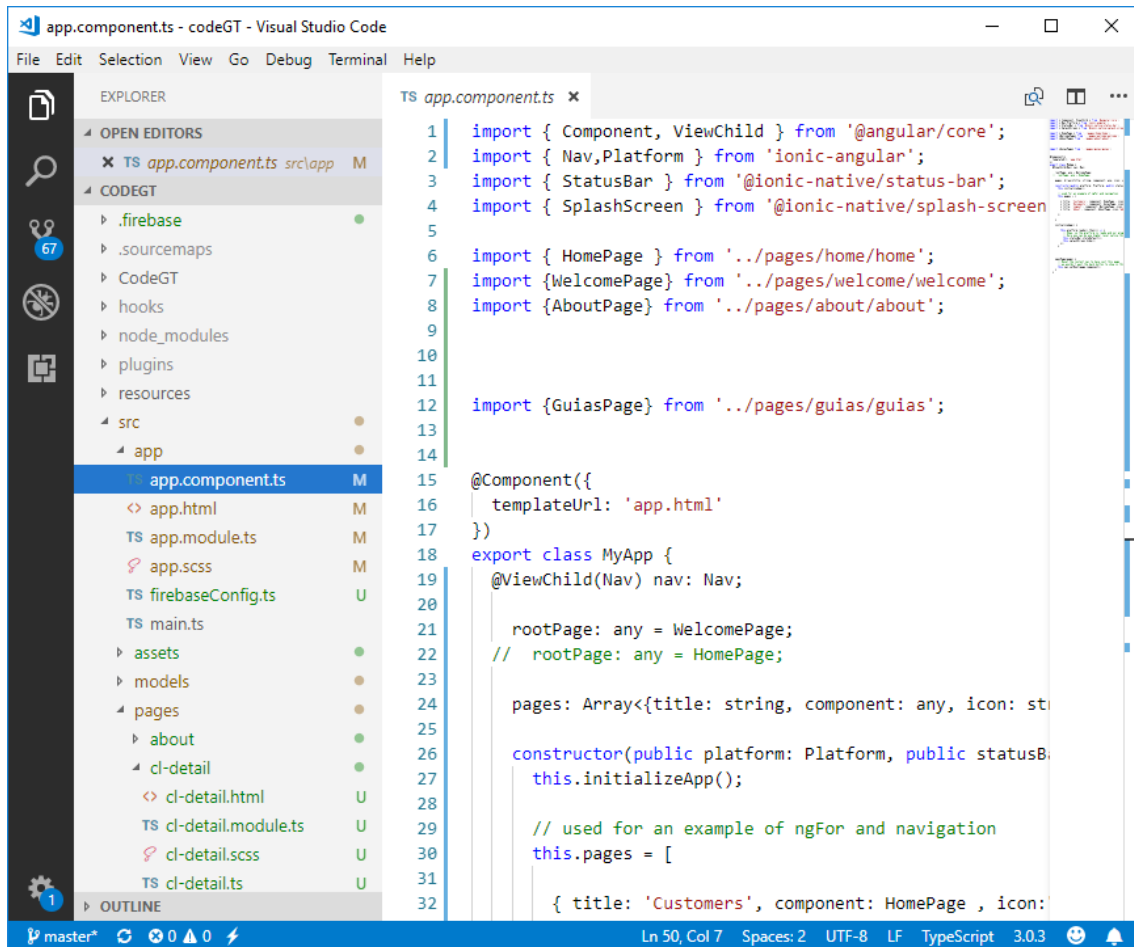


Figure 30 - src directory

The `src/app/app.html` is the main template Figure 31.

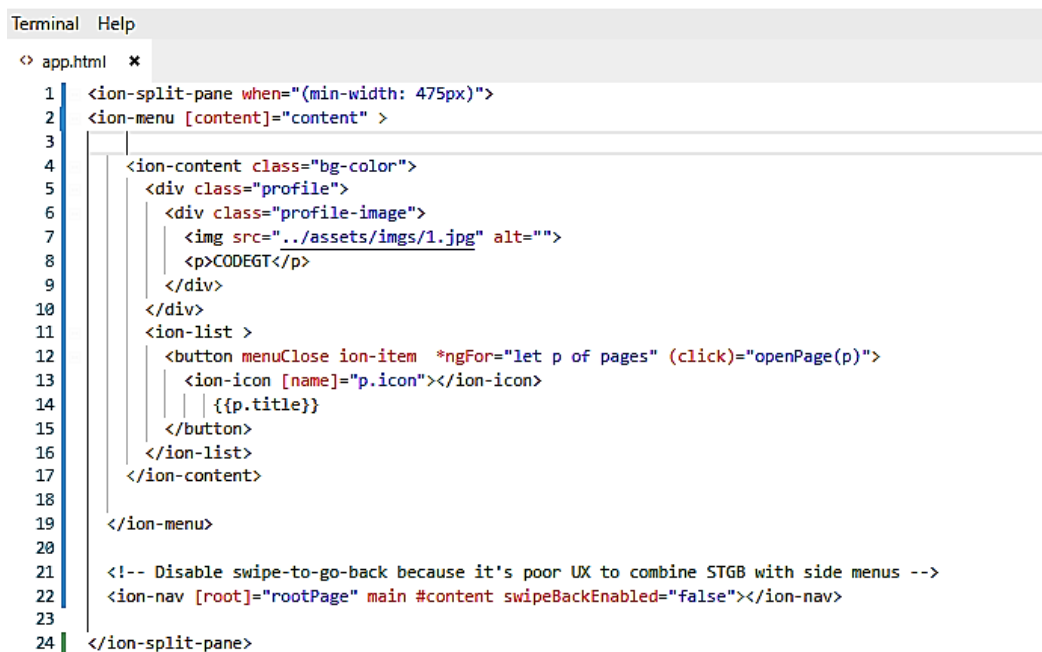


Figure 31 - app.html

To create a new page, one just needs to write on command prompt, as shown on Figure 32.

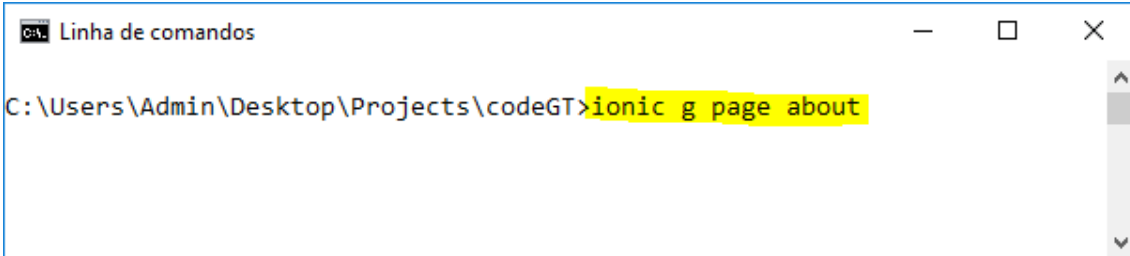
A screenshot of a Windows command prompt window titled "Linha de comandos". The window shows the current directory as "C:\Users\Admin\Desktop\Projects\codeGT" and the command "ionic g page about" entered at the prompt. The command is highlighted in yellow.

Figure 32 - CLI command to generated page about

Ionic CLI will generate the HTML, TypeScript and SCSS files for our new page in a directory under app/pages, Figure 33. The g is for generate, page is for the type of component and about is the name of our new page. With one simple command, a new component was generated for our app (which as the name suggests will be the about page) which contains the following files:

- about.html (the template for our page)
- about.scss (the style rules for our page)
- about.ts (the logic for our page)

Each time you generate a component using the Ionic CLI you will end up with a named directory which contains Sass, TypeScript and HTML files.

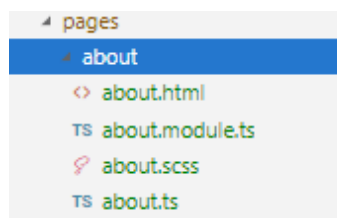


Figure 33 - New page on Ionic

One can see the code for the login Html page Figure 34 and part of the Typescript login page Figure 35.

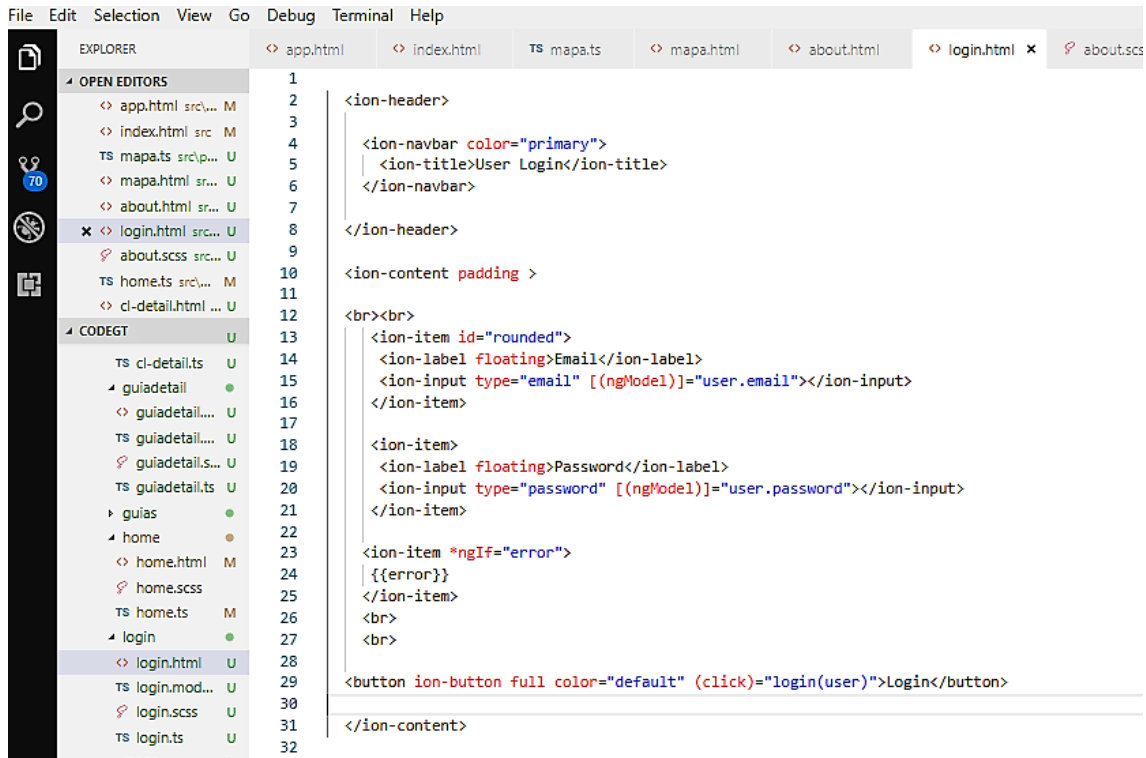


Figure 34 - Html page of Login Page

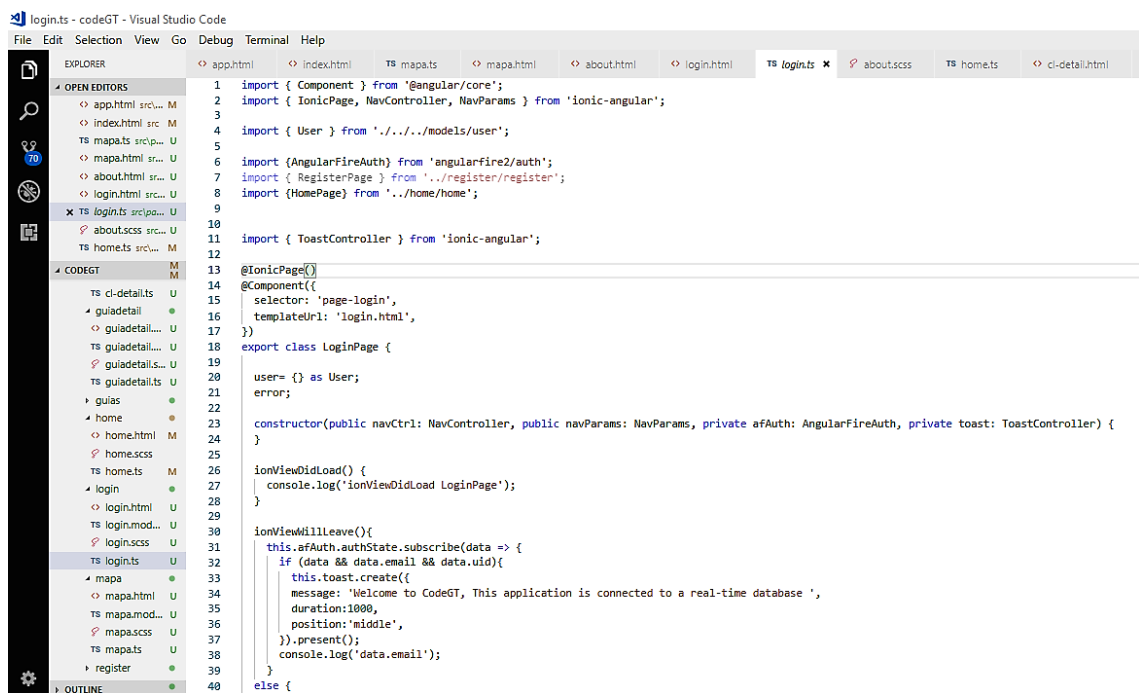


Figure 35 – TypeScript file of Login page

Navigation in Ionic works like a stack of pages, one pushes new pages onto the top of the stack to present to user, or one pops off a page to go backwards Figure 36.

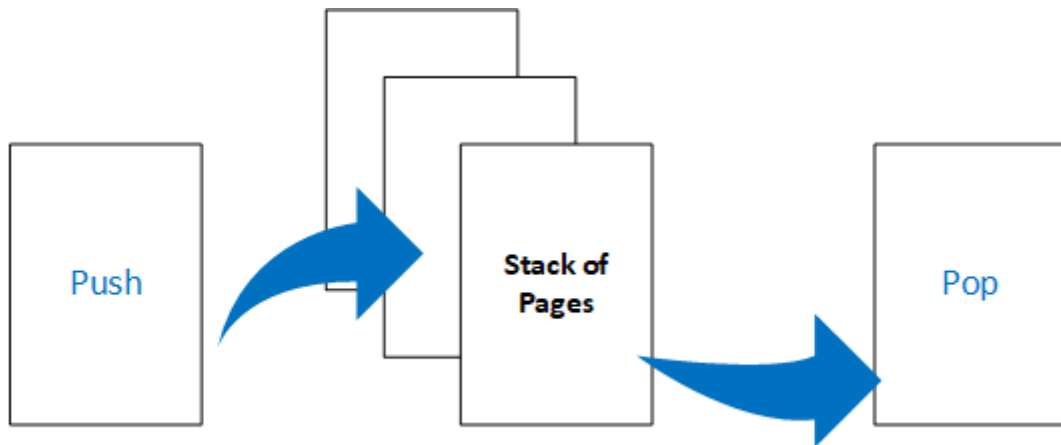


Figure 36 - Ionic Stack of Pages

Ionic framework has building blocks called components. Components allow to quickly construct an interface for our app, and Ionic comes with many components, like buttons, cards, grid, modals, and many more to facilitate our development. To use one component, for example, a button Figure 37;

```
1 < button ion-button color="default">Button</button>
2
```




Figure 37 - Ionic Component - Button

and the result is a blue button with a mobile style.

A three data Model has been created on Ionic platform, one for customers, one for documents (guias) and another one for users. On Figure 38, the guias model is displayed.

```

1  export interface Guias {
2
3  nmdoc?: string;
4  fno?: number;
5  fdata?: string;
6  nome?: string;
7  no?: number;
8  morada?: string;
9  local?: string;
10 codpost?: string;
11 telefone?: number;
12 tlmvl?: string;
13 u_codigoat?:string;
14
15
16 }

```

Figure 38 - Data model on CodeGT

The application uses Authentication, Hosting and Cloud Storage on Firebase services.

a) Firebase

Firebase is comprehensive mobile development platform. Firebase is built on Google infrastructure and scales automatically, for even the largest apps <https://firebase.google.com/>. Firebase platform is a set of products on the cloud for the development of web apps and mobile apps. Google released Firebase in the summer of 2016. Its goal is to provide the tools and infrastructure that you need to build great apps. A set of tools like authentication, database, storage, hosting, analytics and others, everything on cloud services. It's not a replacement for your existing APIs for building Android, iOS, or Web apps. It's an enhancement, giving you common services that you might need – such as a database back end, secure authentication, messaging, and more (Moroney, 2017).

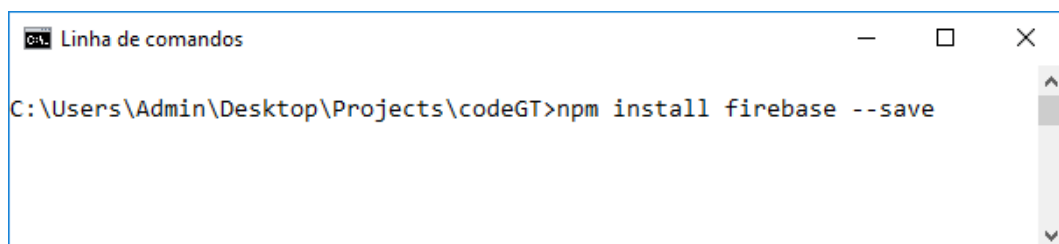
The services are paid, but there is a Spark Plan that is free, that will be used for the development of the mobile application. A Firebase account was created at <https://firebase.google.com> and get a API key for the development. The Authentication API was used for the secure sign-in in the application, the Database, a cloud-based, real time data storage platform and the Storage to register the signatures of the receiver of the goods, in the app. Also, the Firebase Hosting was used to host the mobile application. On reality the users can use the mobile application as an app in the mobile device or can access the application on any device through a web browser.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. It provides syncing across connected devices and is available when there is no network connectivity through a local cache. It is an event-driven database that works very differently from traditional SQL databases. There's no server-side code and database access tiers; all coding is done in the client. Whenever data changes in the database, events are fired in the client code (Moroney, 2017).

Enter the Firebase JS library which consists of the following key services:

- firebase-app - The core firebase client
- firebase-auth - Firebase Authentication
- firebase-database - The Firebase Realtime Database
- firebase-storage - Firebase Storage
- firebase-messaging - Firebase Cloud Messaging

To install the Firebase JS library, one would simply run the following command at the root of our Ionic project on command line utility Figure 39 (I.e. Terminal in Mac OS X or Command Prompt in Windows, the platform create automatically the necessary changes in files for us.



```
C:\Users\Admin\Desktop\Projects\codeGT>npm install firebase --save
```

Figure 39 - Firebase Install

To access the app the user should login. The validation is made by the Firebase API Authentication. After verifying the login, the user can proceed to enter and access the application. To access to the API the user must have a valid login. The application accesses the API from HTTPS to a more secure layer.

One must create a user to login to firebase and then create a project. On Figure 40 it is possible to see the Firebase console and all the functionalities available, like Authentication, Real Time Database, Storage, Hosting and others. For the CodeGT was necessary to create the user's authentications for the login validation of the app, the storage for save the receive signature of the destination customer and finally the Hosting service for host the application.

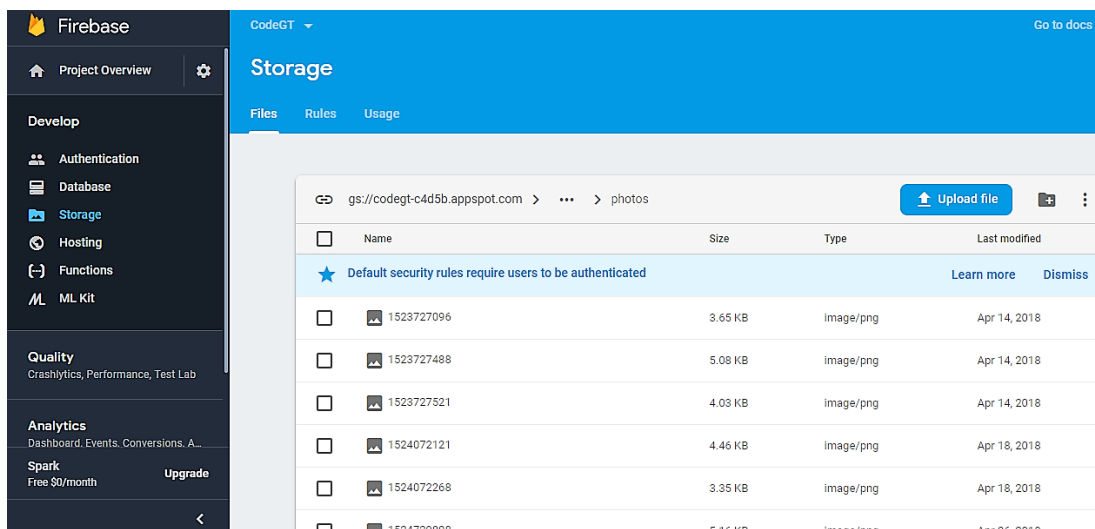


Figure 40 - Firebase Console

The Firebase API Authentication was used for security reasons. So, there is no need to handle the security issues of the users and passwords, delegating this issues to Firebase API. The Ionic platform allows PWA, so the users can not only access the app from the Android Package (APK) and iOS App Store Package (IPA) install to the device but can also access the app through any browser on any equipment desktop or mobile. Accessing the app this way loses the smartphone Native functionalities like phone and GPS.

b) Data Access in the application

Services or Providers can be used by components inside Ionic for providing data or any type of service like;

- HTTP requests;
- Interacting with databases.

In the development of CodeGT app a provider was created to access the API to entities customers and documents of the ERP Figure 41.

```

1  import { Injectable } from '@angular/core';
2  import { Http } from '@angular/http';
3  import 'rxjs/add/operator/map';
4  import { Cliente } from '../models/cliente';
5  import { Observable } from 'rxjs/Observable';
6
7  @Injectable()
8  export class Erpservice {
9    customers:any;
10
11   private urlPage: string = "http://";
12   constructor(private http: Http) {
13
14   }
15
16   getPosts(): Observable<Cliente[]> {
17
18     return this.http.get(this.urlPage)
19       .map(res => res.json());
20
21   }
22
23   postdetails(id) {
24     return this.http.get(`url${id}`)
25       .map(res => res.json());
26
27   }
28
29
30
31
32
33

```

Figure 41 - Provider in Ionic

To create and deploy ipa and apk a MAC computer was needed and run the commands on Ionic CLI:

- ionic cordova build android --prod --release, for build apk for Android devices Figure 42.

or

- ionic cordova build ios --prod, for build ipa, for iOS devices Figure 43.

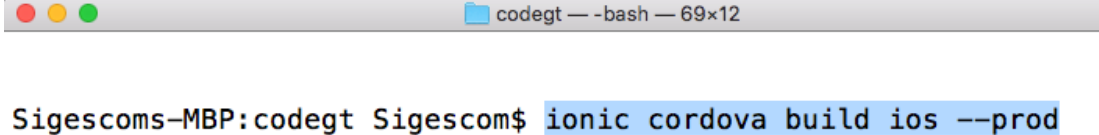
The platform creates on platforms folder each file.

```

C:\Users\Admin\Desktop\Projects\codeGT> ionic cordova build android --prod --release

```

Figure 42 - CLI to build apk



```
codegt — -bash — 69x12
Sigescoms-MBP:codegt Sigescom$ ionic cordova build ios --prod
```

Figure 43 – CLI to build ipa on a MAC computer

c) User Interface

The UI was designed to simplicity, speed, ease of use and fast access to the information, considering that, the access will preferably be through a mobile device. The application starts with a splash screen Figure 44, focusing on the theme of the application. Ionic platform uses components and styling each component to each OS.

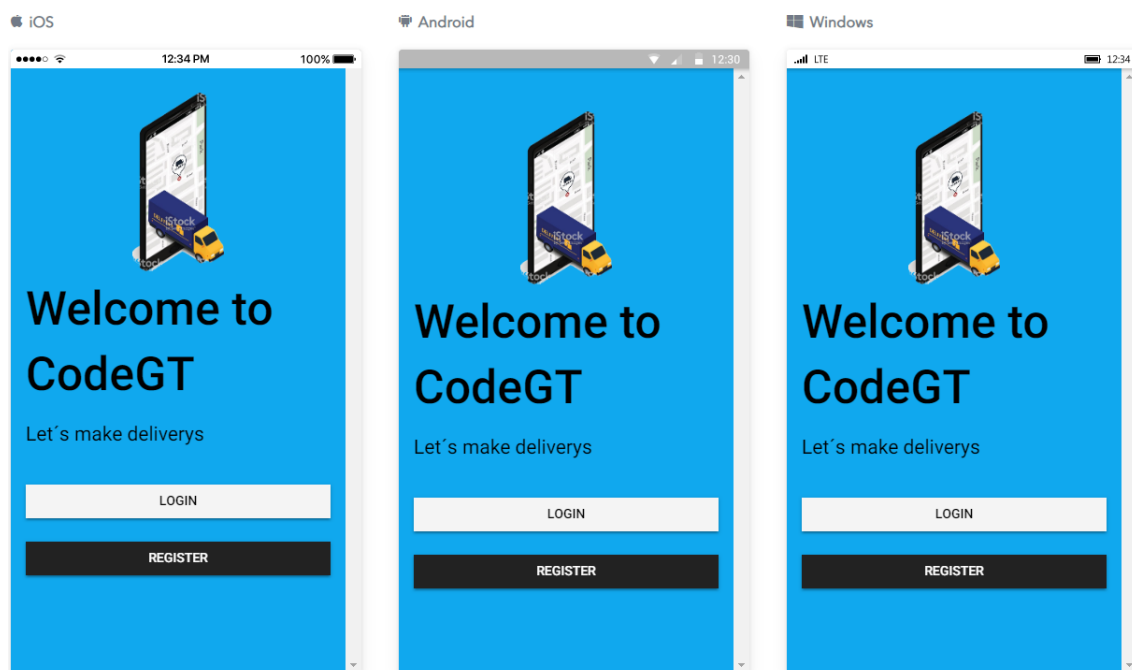


Figure 44- Splash screen on iOS, Android and Windows Phone

To access the app, the user should login (Figure 45). The validation is made by the Firebase API Authentication. After verifying the login, the user can proceed to enter and access the application. The application accesses the API from HTTPS to a more secure layer and only if the user validation is valid.

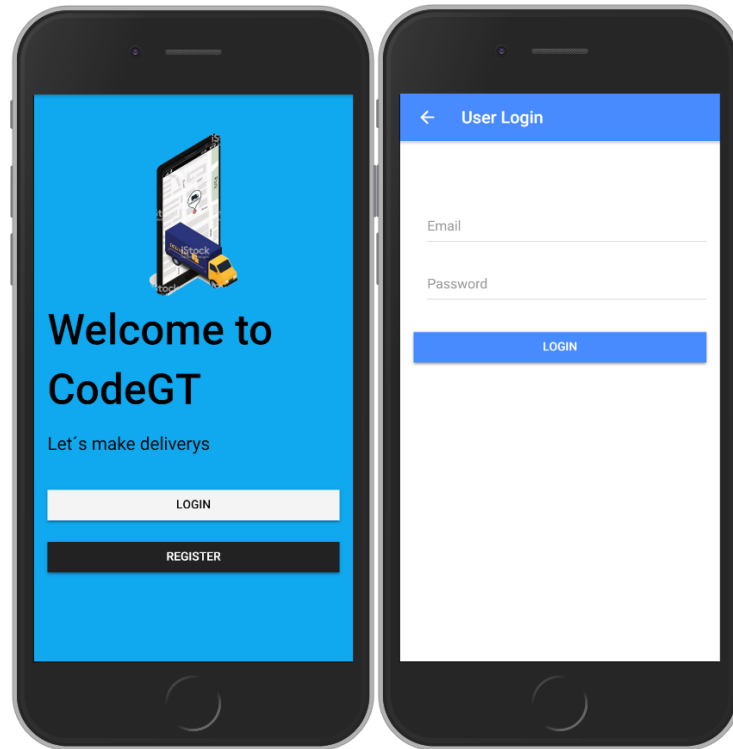


Figure 45 – The Welcome and Login Screen

Access to customer information is very important. The application shows the list of customers (Figure 46 and Figure 47). There is a search field where the user can search for a specific customer by entering at least 3 characters. Services or Providers can be used by components inside Ionic to provide data or any type of service like;

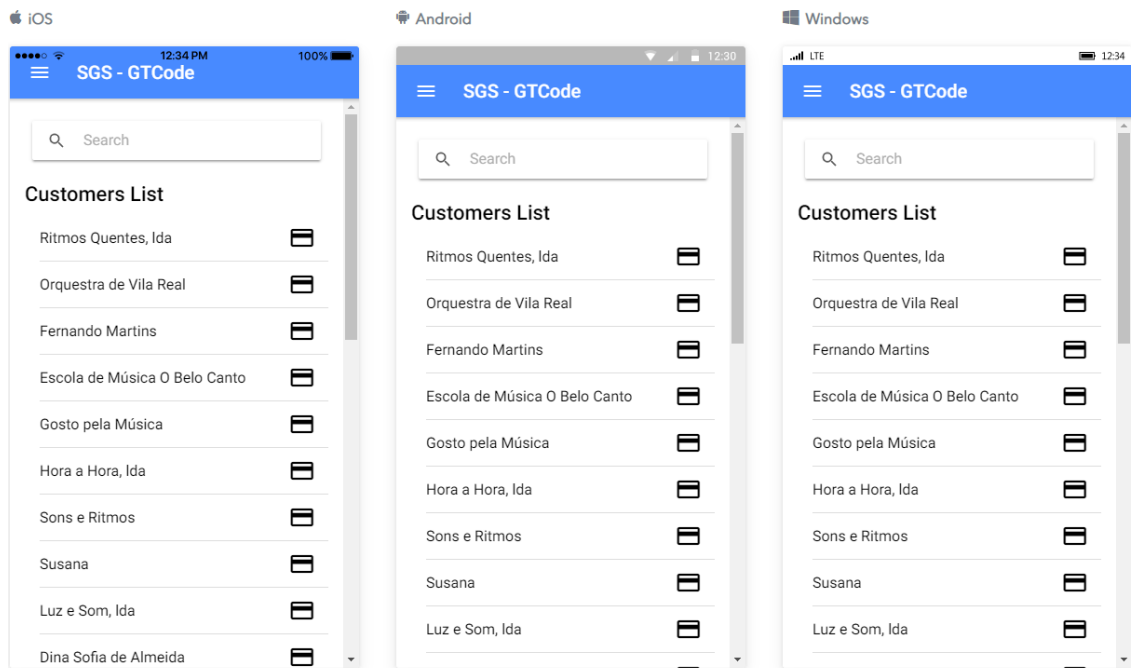


Figure 46 - Customers List on iOS, Android and Windows Phone

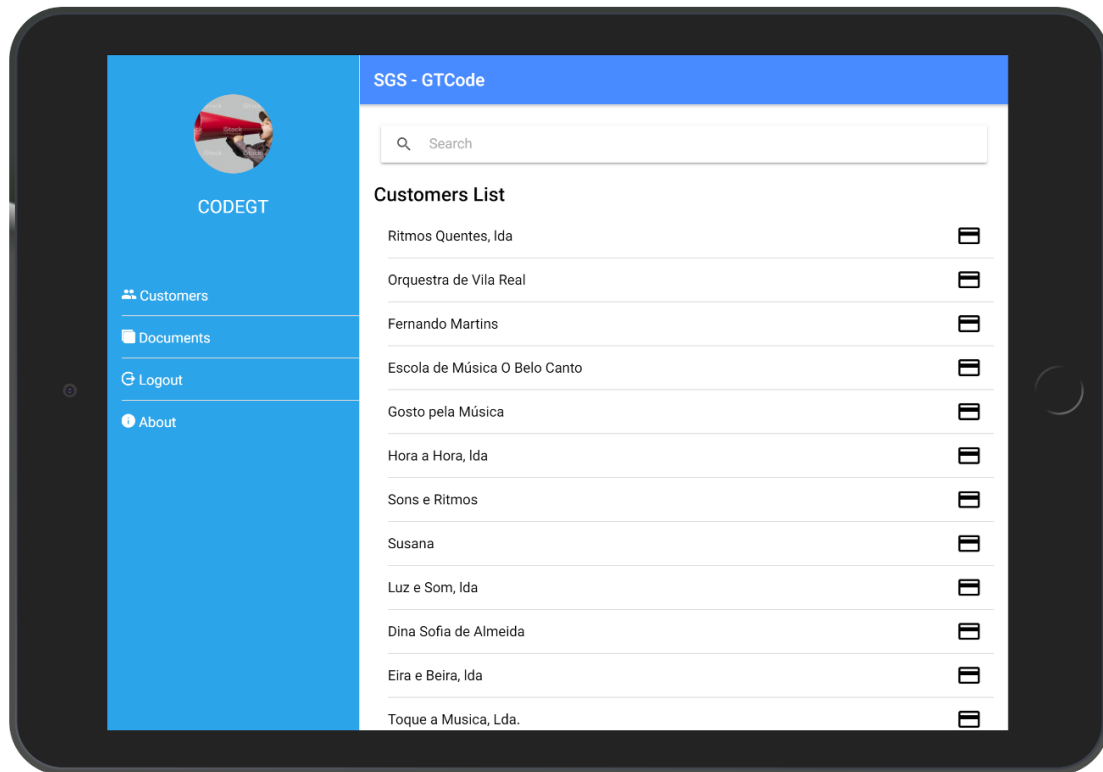


Figure 47 - Layout view from a tablet

After customer selection, a detailed customer account screen appears. You can view the location of the customer's address when selecting a map or call the customer's phone contact by pressing the phone icon in the right corner of the screen Figure 48.

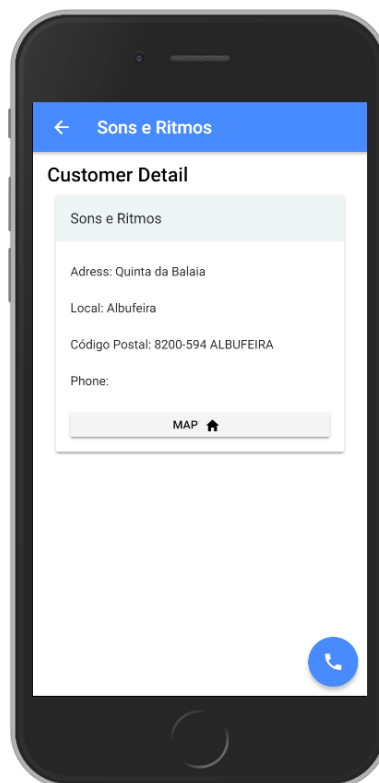


Figure 48 - Customer detail

The menu screen in the upper left corner of the screen shows us other application features, such as, access to documents issued and not yet delivered Figure 49.

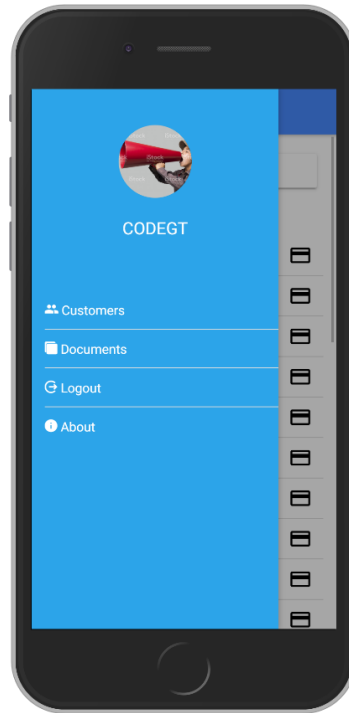


Figure 49 - The Menu screen

Once the function “Documents” is selected, a list of Transport Documents issued and not yet delivered is displayed, through the API Figure 50.



Figure 50 - Pending documents

When the document is selected, the detailed information about the document appears on the screen, with the information about the customer, delivery place and mandatory AT Code Figure 51. This screen is the core screen, with the main information, because here there is the customer's name, the address, the phone number, and the AT code. Thus, one can request the signature of the person who received the goods in the delivery act or call it if there is any need to contact the receiver.

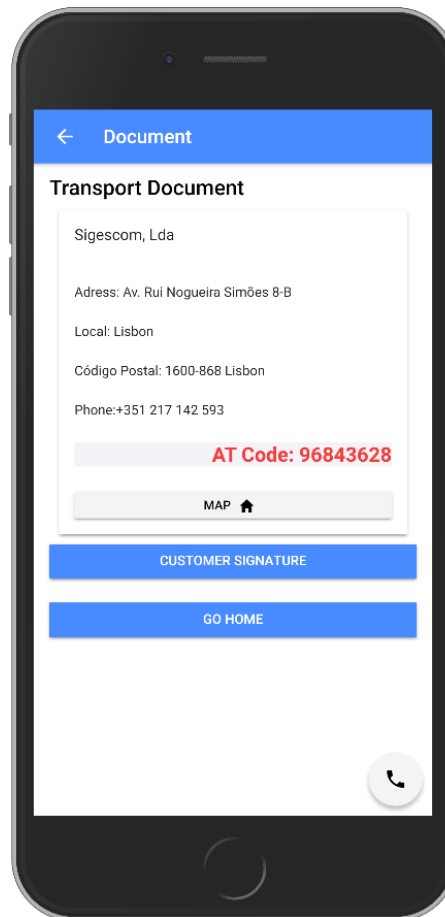


Figure 51 - Detail document

On the map button the user can view the route to destination Figure 52.

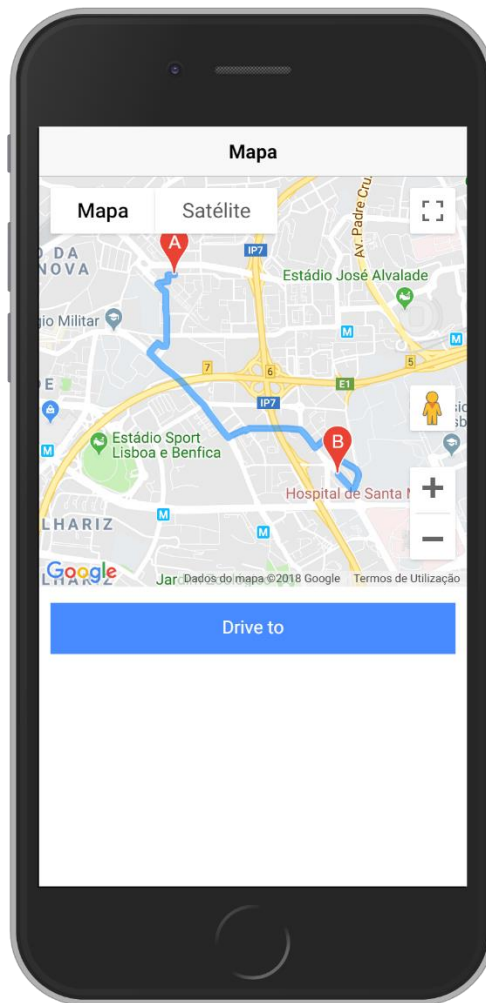


Figure 52 - Map route to destination

On the receiver signature button, the person is requested to sign directly on the app. This process not only registers that this individual received the good and the application will trigger the information that this merchandise has already been delivered, being registered as delivered. The signature is saved in Firebase Realtime Database Figure 53.

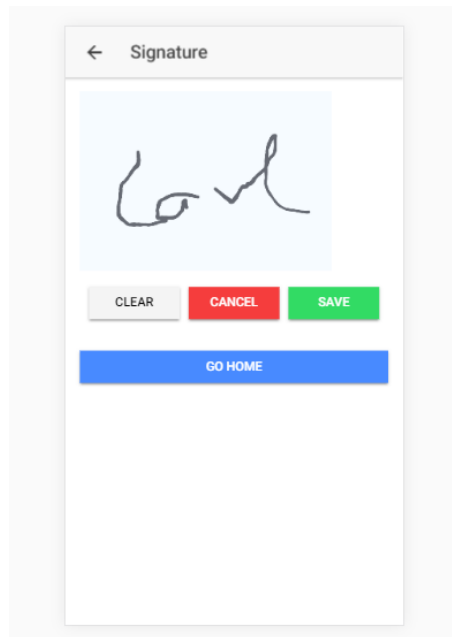


Figure 53 - Receiver signature

When the customer's signature is registered the app returns to the document's screen with the document information and the signature of the receiver Figure 54.

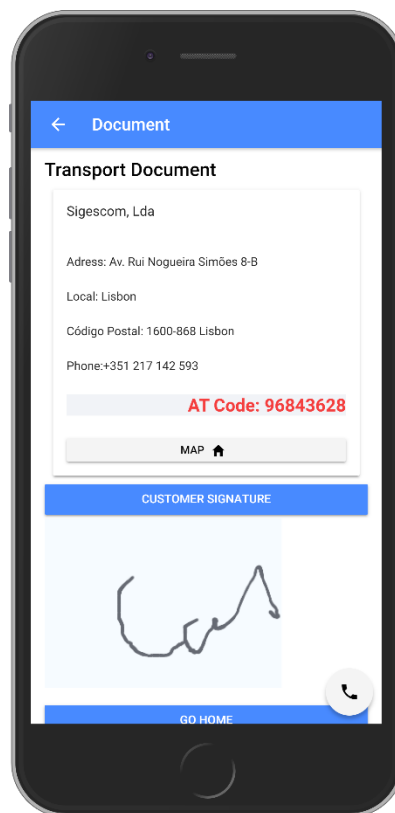


Figure 54 - Document delivery and sign

The about page, shows information about the app and version number Figure 55.

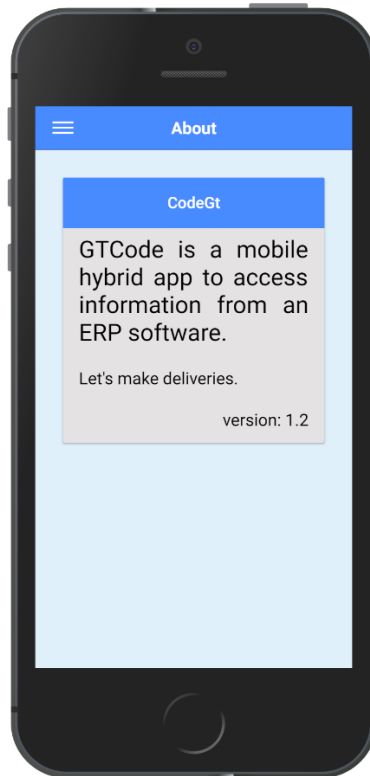


Figure 55 - About us screen

Chapter 4 – Results \ Evaluation

4.1. Qualitative results of the prototype

A prototype implementation of the mobile app in organizations for testing and evaluation was conducted, as CodeGT got the interest from real market industry companies. For that purpose, the mobile app CodeGT was tested on a **real** environment, on two **real** companies. For confidentiality reasons the companies' name were omitted. Company A is an industrial company that delivers its products to final customers and transports material from the factory to a company delegation on a different place. Company B is a group of electronics retail stores with two technicians that transport and install equipment at customer's home. The app was tested by two of each companies' employees.

The sets of tests were carried out through the qualitative evaluation of the solution and a survey of improvements. The app was evaluated on a scale 1 to 5 regarding, the usability, functionalities and performance.

The evaluation on Table 4 was very positive, enhancing ease of use and performance. There is an agreement on the added value that the proposed solution would bring to the current process. Regarding the functionalities, some improvement could be done.

Table 4 - Evaluation mobile Hybrid app CodeGT

Company	User	Usability	Functionalities	Performance
A	JC	5	4	5
A	JB	5	4	5
B	LH	5	4	5
B	RM	5	4	5

The matrix of strengths, weaknesses, opportunities and threats (SWOT) Table 5, shows that the integration with the existing ERP software is a must and ease of use suitable to mobile device with basic information and practical. Thus, it was concluded that the use of mobile devices in the mobility of business systems, like smartphones or tablets, shows great potential for evolution. Through this solution the process was simply digitized and quite practical and productive.

Table 5 - SWOT Analysis

Strengths	Weakness
<ul style="list-style-type: none"> • Low cost • Ease of use • ERP integration • Native device functionalities 	<ul style="list-style-type: none"> • Internet dependency
Opportunities	Threats
<ul style="list-style-type: none"> • Integrated more information from ERP on mobile devices 	<ul style="list-style-type: none"> • PHC ERP launch an app similar or better

Some improvements are necessary and should be complemented on a new version of the app, regarding more functionalities. Suggestions such as the use of the camera in the act of delivery to register and prove how it was delivered without any defect and the use of bar code to read labels on the packages.

Comparing actual process Figure 56 shows the access of ERP from a desktop, or accessing the ERP from a web page Figure 57 and from the mobile app. The CodeGT app was well accepted, practical

Figure 58, clean and well suitable to mobile devices.

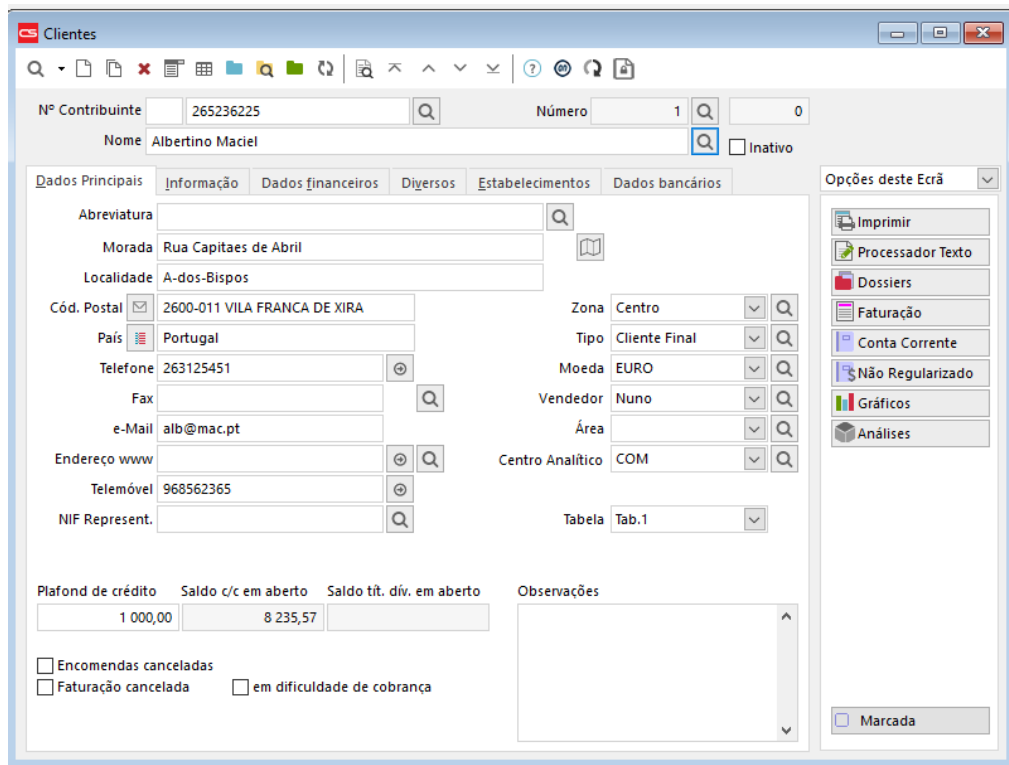


Figure 56 - ERP PHC desktop UI for a specific customer

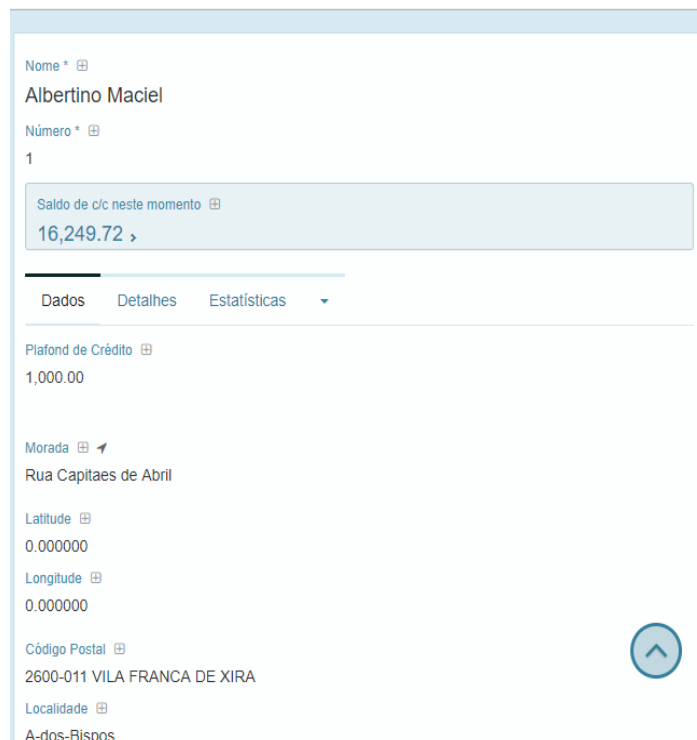


Figure 57 - ERP PHC Web UI for a specific customer

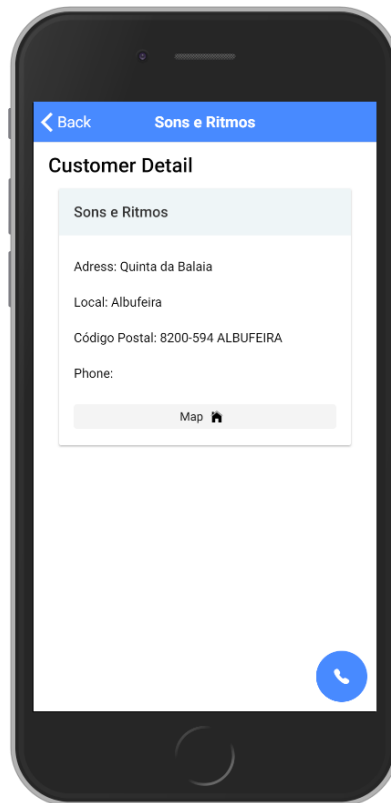


Figure 58 - CodeGT UI for a specific customer

4.2. Quantitative Evaluation

4.2.1 Cost for developing a mobile app

The average cost for developing a mobile app depends on several factors, some considerations were made regarding the costs involved on mobile software development:

- App functionality and purpose;
- Visual design UX and UI;
- Customization;
- Mobile platforms and devices supported;
- Use of Native features of the devices;
- Backend infrastructure, Server-side, hosting if needed;
- App administration;
- Geographical location;
- Maintenance.

All these factors influence the cost of mobile application development. If the target is a basic app with few functionalities the cost will be lower, but if a complex app is needed, more hours will be necessary to all the life cycle of the development of the app.

The User Experience (UX) and User Interface Design (UI) are important, User Experience Design, refers to the design of an application's usability, accessibility, and interactions, to enhance users' satisfaction when they operate a system, while UI is the design focus on maximizing the usability and the UX. Ionic components maximize visual design for multi-platform since Ionic components are automatically styling for each platform so has a best UX across platforms.

More customization means more time and less code reuse, implying more costs. On mobile app development some UI is needed for each device and platform, so more time is spent to style the app for more devices. If the target is a multi-platform app, more development hours are needed on a Native approach. Backend infrastructure, servers, API access, cloud storage or hosting will make the app development costs rise. If the app needs an administration page, more development time is needed. It is known that a developer hourly cost is different from country to country, so the cost also depends on where the app is developed. If maintenance or upgrades are needed, it obviously involves more costs.

Considering one mobile app project with a single platform target, on Figure 59 one can see higher costs on Hybrid development through development life cycle because Native device functionalities are intrinsic on Native OS, while the Hybrid platforms must work around to get it. Also, the rewrite code on Native OS are more common. Usually, for creating apps with the access to mobile devices new release features, platform-specific APIs are used. With the emergence of new features, there is a need of introducing new APIs, against which the developers haven't yet developed. Thus, developing time increases, which affects the total cost.

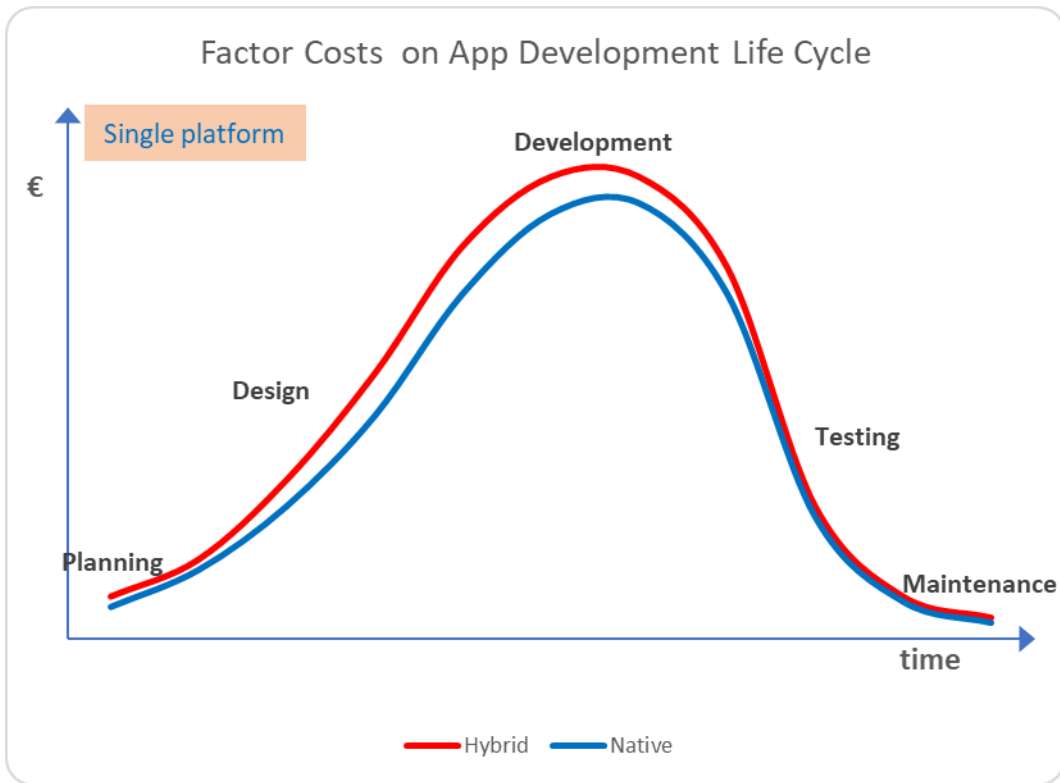


Figure 59 - Costs on App Development Life Cycle for Single Platform

For one project with a multi-platform targeting the Hybrid cost becomes less expensive since one does not need to develop several code-base to reach several platforms Figure 60.

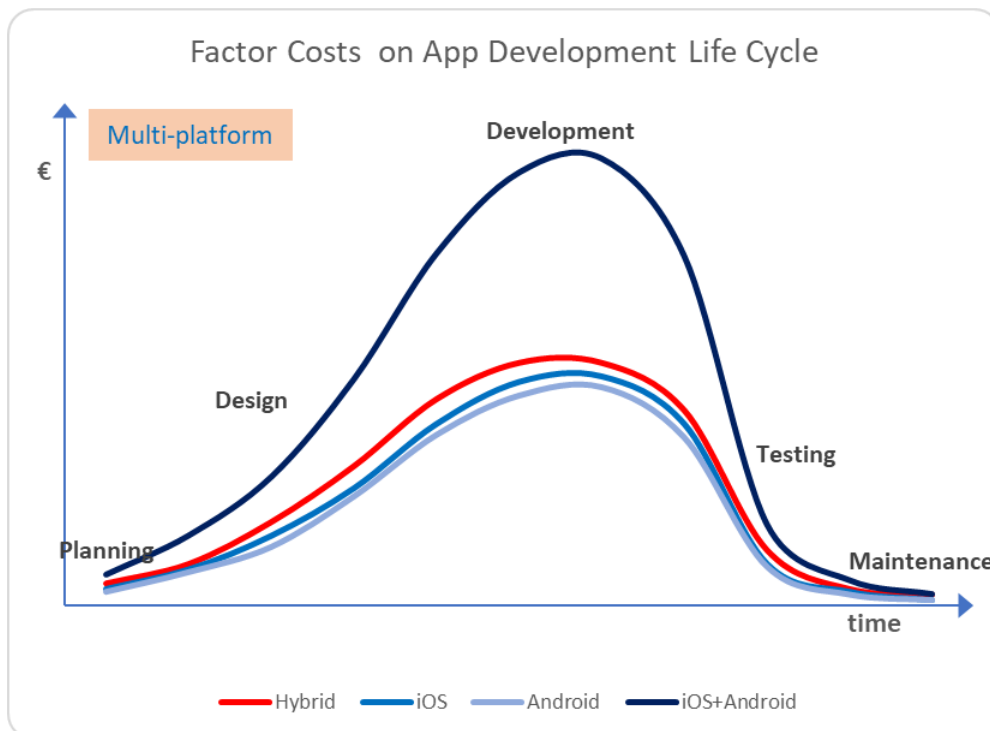


Figure 60 - Costs on App Development Life Cycle for Multi-Platform

On the projects perspective Figure 61, the costs involved on single project for one platform development, the Hybrid development is high due to the long-time need it to learn all the programming language, technologies and platforms involved.

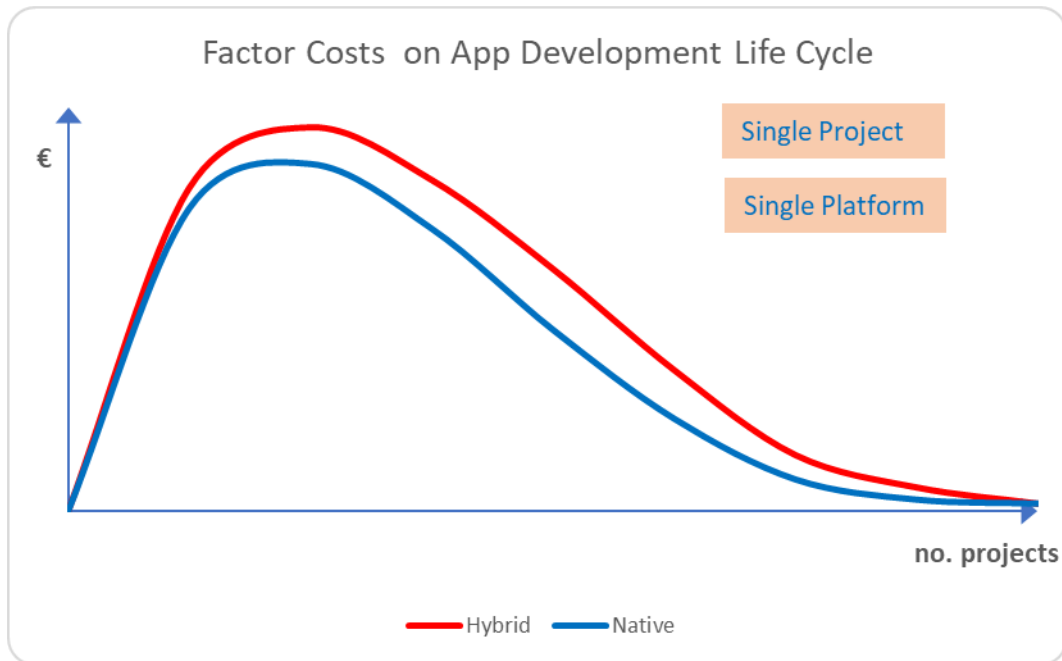


Figure 61 - Costs on App Development Life Cycle Project for Single Platform

The costs involved on several projects targeting multi-platform, the Hybrid development is cheaper due to the initial cost because the learning curve will be diluted on the several projects Figure 62,. If the target is to develop a mobile app, targeting multi-platform for iOS and Android apps, a Hybrid app development is a good choice in terms of cost. The Native problem is that usually you need to have a separate team for each platform and the app creation cost is almost doubled. If an app is Hybrid, you have only one team working on it.

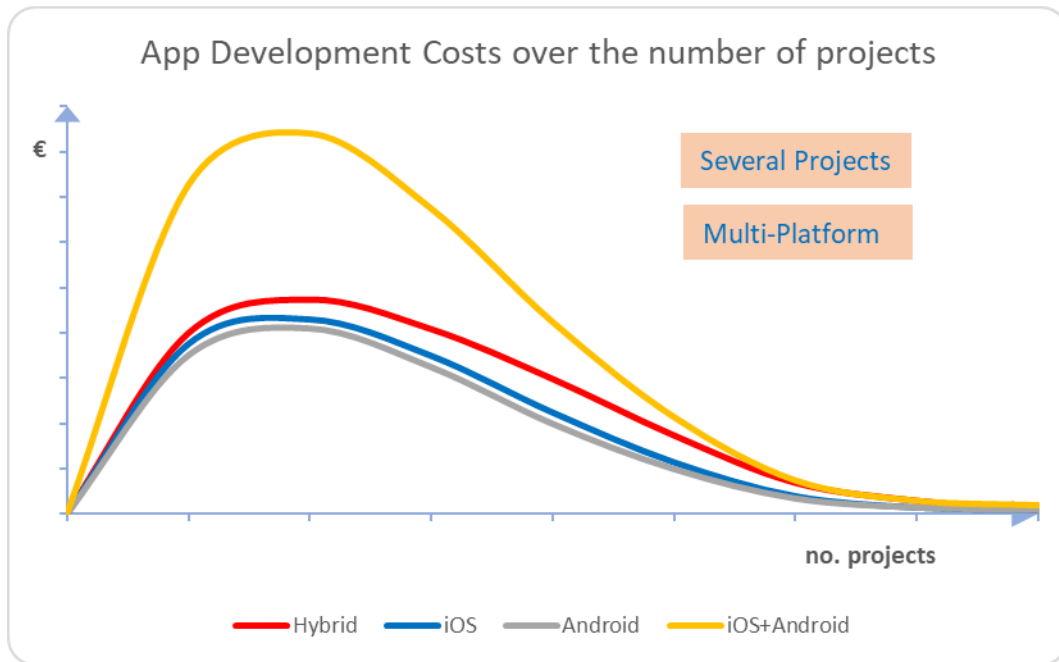


Figure 62 - Costs on App Development Life Cycle Project for Multi-Platform

It must be assumed that app life cycle could start as a Hybrid app, because of lower costs and faster development time and if it will become a successful app, then there will be financial resources to invest more to a multi-platform Native app approach, increasing complexity and performance, or continue on Hybrid development increasing app complexity. So, the risk of investment will be lower to the beginning until it reaches the return of investment point.

4.2.1 App Complexity

An app complexity scale classification has been created based on moderate time needs to develop on Figure 63.

- Basic app - an application with simple functionality that requires approximately 300-700 hours on development.
- Medium complexity app takes from 700 to 800 hours.
- Complex time-consuming app in most cases exceeds 1200 hours on development.

Giving a rough estimate of application development cost (taking the rate of 35€-50€ an hour on average based on past experience): a basic application will cost around 7 500

€ - 20 000 €; medium complexity apps will start from 20 000 € to 75 000 €; the cost of complex apps usually goes beyond 75 000€.

Apps Complexity Scale

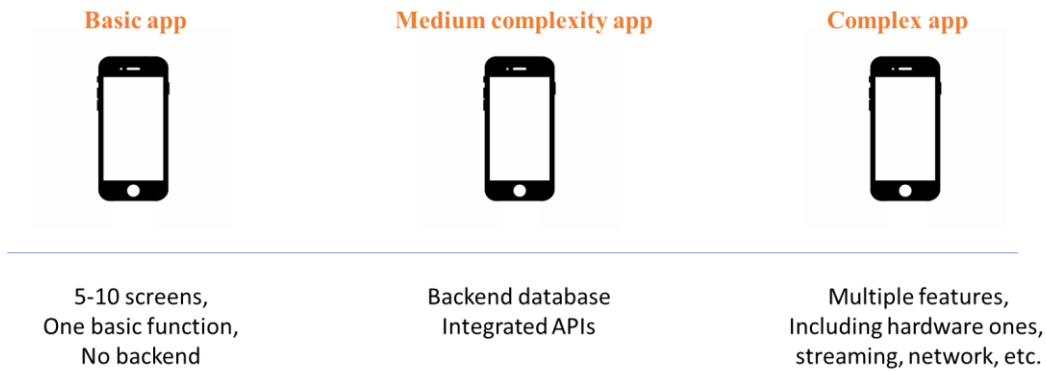


Figure 63 - Apps Complexity Scale

The Hybrid development approach is used more on a basic or medium app and Native for more complex apps. If a more complex and demanding application is the focus, usually a Native development is used. The benefits for choosing a Native mobile app, includes the assurance of performance, precision, and perfection. As a developer, there is no concern about any bugs or lags that might arise due to an unforeseen incompatibility with the operating system. Consistency is guaranteed as the functioning of the application is in sync with the other primary applications hosted by the OS.

The number of apps in the app store reaches more than 5 Million apps. Where Google Play Store has almost double the number of apps of the Apple App Store Figure 64.

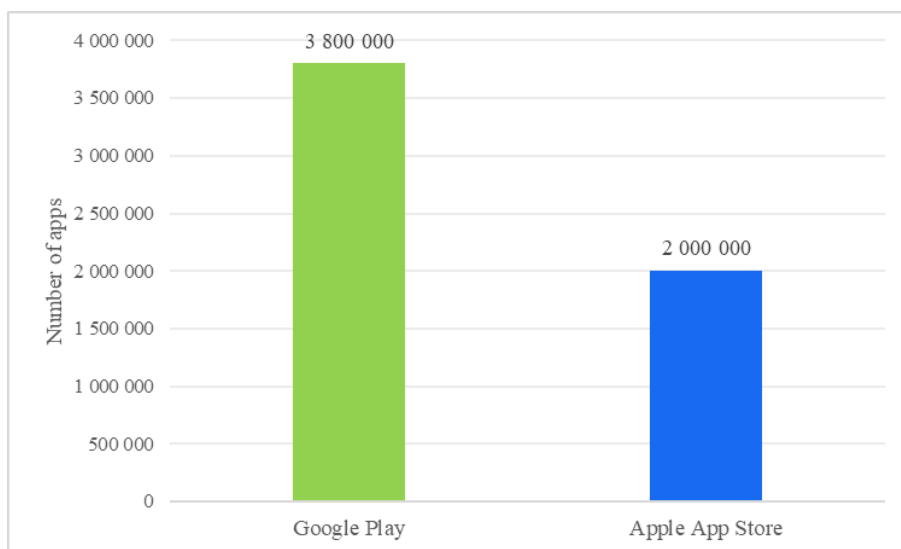


Figure 64 - Number of apps available in leading app stores as 1st quarter 2018(Statista, 2018)

A huge gap exists between free apps and paid apps. Although the Google Play Store has more apps the Apple Store has a bigger percentage of paid apps because of the iOS customers profile who are more likely to spend more money Figure 65.

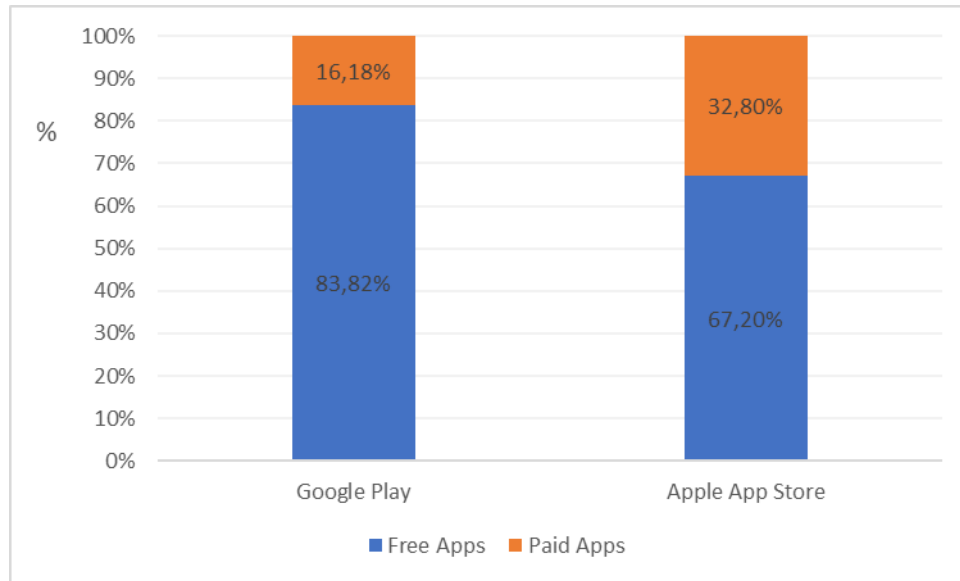


Figure 65 - Free vs. Paid Apps (Statista, 2018)

The revenue of a mobile app Figure 66 comes especially from;

- Paid value on client app acquisition
- Advertising on the app
- In-App Purchase (IAP). In-app purchases includes any transactions performed in-app (a virtual goods purchase where the app store takes a cut, example, booking a hotel in a travel app, etc.).

This means that even basic or medium apps can have revenue from advertising and in-app purchases. On a point of view from software company developers, an initial lower investment in the development of mobile apps with Hybrid platforms and still get good revenues without being a paid app on App Stores.

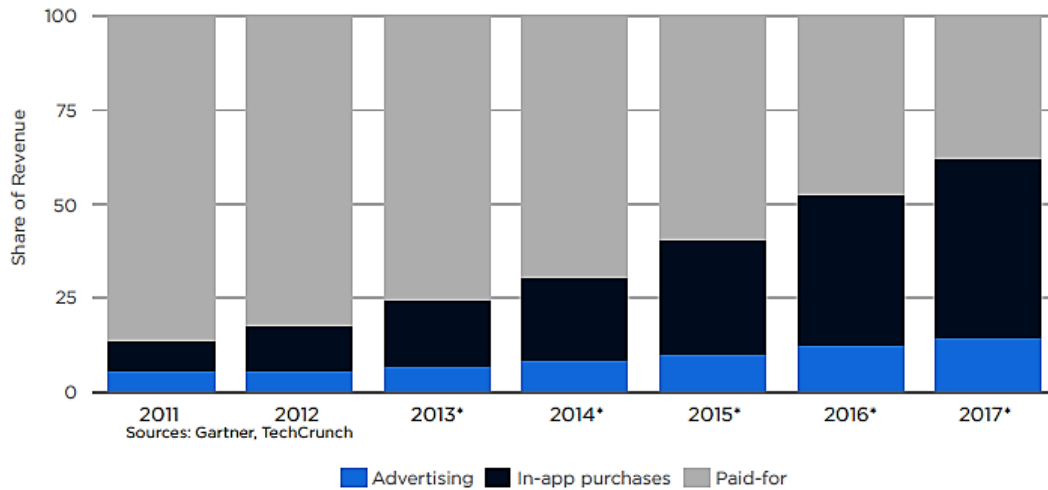


Figure 66 - Share of Global Mobile App Revenue By Type (Appsflyer, 2016)

According to Pordata Figure 67 and Figure 68, there was 1 213 107 SME in Portugal in 2016 that represented 99,9% of the total companies. Eurostat defines that enterprises are classified in different categories according to the number of people employed. SMEs stand for small and medium-sized enterprises with fewer than 250 persons employed. SMEs are further subdivided into:

- micro-enterprises: fewer than 10 persons employed;
- small enterprises: 10 to 49 persons employed;
- medium sized enterprises: 50 to 249 persons employed.

The vast number of SMEs companies is also a great opportunity for Hybrid mobile development, since SMEs usually have less capital to invest in a mobile app.



Figure 67 - Number of SMEs in Portugal in 2016 (Pordata, 2016b)

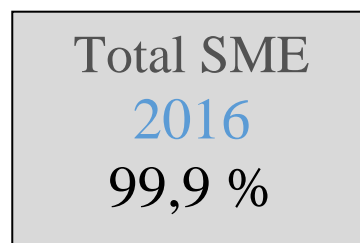


Figure 68 – Percentage of SMEs in Portugal in 2016 (Pordata, 2016a)

If mobile apps start as free, one can estimate that there is a huge market for Hybrid mobile apps, for start-ups or small software companies to develop more and cheaper multiplatform apps using Hybrid development for the huge market of SMEs. As apps become successful, one can assume that there is then money to invest in app development, increasing complexity and moving to Native approach.

4.2.2 Mobile Hybrid Development Risks

There are long-running challenges facing software developers. On Figure 69, it is possible to see that many of them are regarding training needed to acquire and to get on continuing upgrades on dynamics platforms in immature tools.

Through the text, I have already mentioned the risk of the Ionic platform during the time of this dissertation on several upgrades due to Angular and Firebase upgrades and from updates of the platform itself. Some of them were disrupted, meaning some rewrite code was needed. Hybrid platforms have evolved a lot in recent years to improve their approach to the performance and functionality of Native but there is the risk of breaking code.

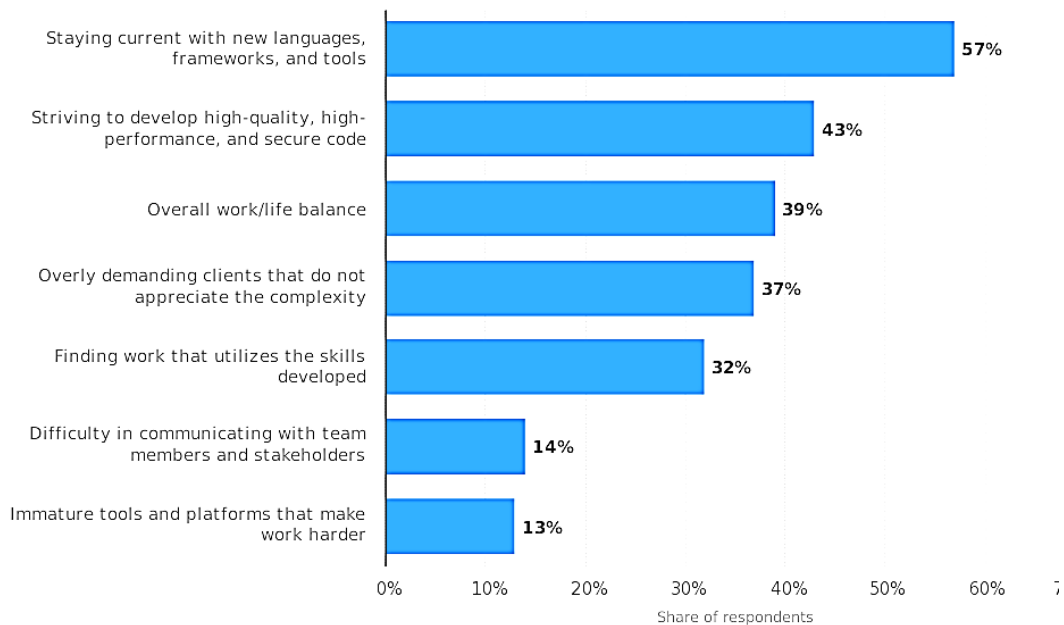


Figure 69 - Most pressing long-running challenges facing software developers worldwide as of 2015(Statista, 2015)

4.3. Analysis and discussion of results

The learning time needed for all the technologies and languages in the CodeGT app development platform was long. A long period of time was needed for learning all the programming language and technologies involved in the CodeGT mobile Hybrid app development, like C#, JavaScript, TypeScript, HTML5 and CSS, Angular, Cordova and the Ionic Platform. Even so, the time was less than the necessary to learn all the programming languages and tools for the development of the application in the two-main Native operating systems IOS and Android. With the advantage that the application can be deployed in any mobile platform, any App Store and can still be accessed through any device via browser, via hosting (but without Native functionalities in this approach).

The advantages are that the result is a mobile Hybrid application that work on the most popular mobile operating systems and platforms and run as well as a Progressive Web App (PWA), so also run in any web browser in any device mobile or desktop. I estimate that the time was half of the time of the Native approach with the advantages mentioned.

The Hybrid mobile application can be deployed to mobile device or can be accessed through any desktop equipment accessing a site on Firebase hosting through any web browser, which is a huge advantage.

Chapter 5 – Conclusions

5.1 Main conclusions

With the increasing number of mobile devices, there has been a trend away from desktop applications towards mobile apps. Many more mobile applications are needed to interact with clients and end users. The demand for mobile experiences is growing 5x faster than IT teams can deliver (Gartner, 2015). So, there is a delivery gap that Hybrid platforms can help, delivering faster and cheaper apps on the market.

Native applications offer by far the best user experience and can use all Native features and functionalities of the devices. Native offers rich Native libraries and SDKs, better performance, with less 3rd party dependencies. When using Native apps targeting several OS, this means that separate teams are needed to build in parallel, manage multiple codebases, hire specialized and costly Native developers, so a higher cost is a major issue on that approach.

Hybrid and Native apps have been coexisting with a higher performance from Native ones. However, with the rapid development of HTML5 and Hybrid app platforms, the performance of Hybrid apps is improving. The cross-platform mobile development solutions shorten the software development lifecycle by writing the mobile application once to then run it on different platforms. The use of platforms for developing mobile Hybrid apps will increase to maximize the impact of their coding effort by applying the concept of “Write (develop) Once and Run Anywhere” (WORA) (Huynh & Truong, 2017).

Cross-platform mobile frameworks such as Microsoft’s Xamarin, Apache Cordova and Ionic are continuously improving and replacing the use of Native development in Basic and medium complexity apps.

The demand for mobile application development is a reality and more and more mobile app will arise focus on cross-platform with Hybrid frameworks development advantages. The advance in Hybrid framework in general and the growing acceptance of open source frameworks, such as Ionic in particular, may provide an alternative to the Native app domination. With Hybrid framework like Ionic, it is possible to run our app on any platform or device from a single codebase and access device capabilities via plugins. It can even run our apps in a regular browser as a Progressive Web App (PWA), which is a major advantage over Native development.

The cost involving the mobile development is cheaper and faster in a Hybrid approach if a multi-platform app is the target. Hybrid platforms now are more powerful and easier to use, and I believe they will be more and more used by developers.

The combination of mobile app platforms development technologies, backend platforms and, along with new mobile device hardware capabilities, will continue to push mobile apps evolution. This proves hypothesis HYPOTHESIS 1.

The mobile Hybrid application CodeGT, resulted in an intuitive and clear UI application for the user and the access to the API and firebase was quite fast. The developed mobile app shows a UX friendly and faster. Ionic platform is a valid alternative development platform for cross-platform development creating an apps to interact as an add-in for the ERP software. Mobile Hybrid platforms are a good solution to build mobile apps faster and cheaper to interact to an ERP with a good UX Figure 70.

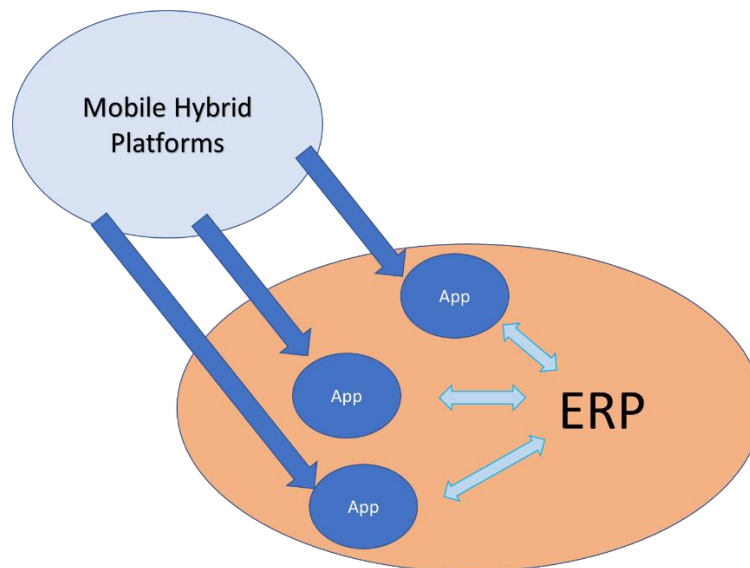


Figure 70 - Hybrid platforms solution to build mobile apps to interact ERP

The Hybrid mobile application CodeGT can be deployed to any mobile device or can be access through any desktop equipment accessing a site on Firebase hosting through any web browser, which is a huge advantage. This proves HYPOTHESIS 2.

Having a Hybrid app framework in place is profitable for business groups, enterprises, and start-ups that are looking to reach out to a greater number of people in a lesser amount of time. On multi-platform target Native apps require more time and money for development, while Hybrids can be hosted quickly and with greater ease and cheaper.

There were some negative points, through the development of the project CodeGT, Ionic platform had constant updates. Some updates were due to Angular updates, others due to the dynamics of the Ionic Team to develop a better Hybrid platform. All these updates required continued training to learn new and disruptive functionalities, what could be seen has a risk.

5.2 Main Scientific and Business Community Contributions

5.2.1 Contributions at the academic level

The research contributions include the analysis of the state of the art in the related fields of mobile app development and ERP software, also the analysis of cross-platforms more specifically Hybrid development approach advantages, versus Native, are clearly highlighted in this document on sections 4.3 and 5.1.

A creation of a proof-of-concept where the proposed mobile app CodeGT using Hybrid platforms.

This research was also supported by peer validation in the acceptance and publication of a paper accepted for publication in the scientific IEEE International Conference of Intelligent Systems 2018 (<http://www.ieee-is2018.com>), where the paper and research were presented to scientific live participation in referenced conference and workshops in the fields of research.

International Conference with double blind Peer Revision:

- Pinto, Carlos Manso; Coutinho, Carlos Eduardo, “From Native to Cross-platform Hybrid Development”, in Proceedings of the 9th IEEE-TEMS International Conference on Intelligent Systems 2018 (Funchal, Madeira, 2018-09-26). To be published.

5.2.2 Contributions at the industry and business level

The development of mobile applications through Hybrid platforms are an opportunity for organizations to create apps intended to all mobiles devices, faster and cheaper than the Native approach. Several mobile Hybrid apps can be developed as modules to interact to any ERP given a very good UX experience to the user. The presented application CodeGT not only was developed but was considered of excellent value for real industry companies which are interested in it.

5.3 Future work

In the last years, many new Hybrid platforms are emerging to give a cross platform approach and I expect more will come up in the years to come to decrease a difference in both Native and Hybrid approaches, also with the evolution with the existing ones in the market.

Developers can use client-side technologies to build client apps themselves, using specific frameworks and patterns for a cross-platform experience.

In a future version of the mobile app CodeGT, I have the challenge of developing new modules that can complement the actual application, maintaining the strategy of complementing the ERP in a mobile perspective. Thus, new modules must be developed to facilitate mobility and access through mobile equipment the information in the ERP. As an example, an agenda with a calendar, tasks and scheduling, or a bar code reading on the confirmation of delivered products. Another approach will be creating analytics with dashboards for managers.

From the market point of view, the mobile app CodeGT, aims all potentials enterprise customers, who use an ERP system and wants to interaction to the ERP from a mobile device.

Bibliography

- Accenture. (2018). (m)apping the future, *Enterprise*.
- Adams, L. G., Persaud, R., Acworth, G., Adams, D. G., & Hamadeh, S. (2013). SPLASSH (Student Programs Like Aquatic Science Sampling Headquarters) <http://splash.meteor.com> a socially driven platform about water. *Oceans-San Diego*, 1–9. <https://doi.org/10.23919/OCEANS.2013.6741344>
- Adinugroho, T. Y., Reina, & Gautama, J. B. (2015). Review of Multi-platform Mobile Application Development Using WebView: Learning Management System on Mobile Platform. *Procedia Computer Science*, 59(Iccsci), 291–297. <https://doi.org/10.1016/j.procs.2015.07.568>
- Alan R. Hevner, Salvatore T. March, J. P. and S. R. (2004). Design Science in Information Systems Research. *MIS Quarterly*, Vol. 28, N.
- Angular - One framework. Mobile & desktop. (2018). Retrieved January 1, 2018, from <https://angular.io/>
- Apache Cordova. (2018). Retrieved from <https://cordova.apache.org/>
- Apps, D. M. M., Justin, J., & Jude, J. (2017). *Learn Ionic 2*.
- Appsflyer. (2016). The State of In-App Spending, 1–22.
- Autoridade Tributaria. (2012). Regime de bens em circulação objeto de transações entre sujeitos passivos de IVA. Retrieved January 1, 2018, from <https://dre.pt/application/conteudo/174543>
- Bahrami, M., Arabzad, S. M., & Ghorbani, M. (2012). Innovation In Market Management By Utilizing Business Intelligence: Introducing Proposed Framework. *Procedia - Social and Behavioral Sciences*, 41, 160–167. <https://doi.org/10.1016/j.sbspro.2012.04.020>
- Barafort, B., Shrestha, A., Cortina, S., Renault, A., Mesquida, A.-L., & Mas, A. (2018). A Software Artefact to support Standard-based Process Assessment: Evolution of the TIPA® Framework in a Design Science Research Project. *Computer Standards & Interfaces*, (April), 0–1. <https://doi.org/https://doi.org/10.1016/j.csi.2018.04.010>
- Bera, M. H. G., Mine, A. F., & Lopes, L. F. B. (2015). MEAN Stack : Desenvolvendo Aplicações Web Utilizando Tecnologias Baseadas em JavaScript.
- Bosnic, S., Papp, I., & Novak, S. (2017). The development of hybrid mobile applications with Apache Cordova. *24th Telecommunications Forum, TELFOR 2016*. <https://doi.org/10.1109/TELFOR.2016.7818919>
- Bott, R. (2014). *Ng-Book. Igarss 2014*. <https://doi.org/10.1007/s13398-014-0173-7.2>
- Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. (2018). Retrieved February 28, 2018, from <https://ionicframework.com/>
- Build Apps with JavaScript | Meteor. (2018). Retrieved February 21, 2018, from <https://www.meteor.com/>
- Charland, A., & Leroux, B. (2011). mobile application Development : Web vs . native. *Communications of the ACM*, 54, 0–5. <https://doi.org/10.1145/1941487>
- Chou, D. C., Bindu Tripuramallu, H., & Chou, A. Y. (2005). BI and ERP integration. *Information Management & Computer Security*, 13(5), 340–349. <https://doi.org/10.1108/09685220510627241>
- Davenport, Thomas H., Prusak, L. (1998). *Working Knowledge*. Harvard Business School.
- Davenport, T. H. (1998). Putting the Enterprise into the Enterprise System. *Harvard Business Review*, 1–12. <https://doi.org/Article>
- Eisenman, B. (2015). *Learning React Native: Building Native Mobile Apps with JavaScript*. O'Reilly Media. Retrieved from

- <https://books.google.com/books?id=t74fCwAAQBAJ&pgis=1>
- El-Kassas, W. S., Abdullah, B. A., Yousef, A. H., & Wahba, A. M. (2017). Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal*, 8(2), 163–190. <https://doi.org/10.1016/j.asej.2015.08.004>
- Elbashir, M. Z., Collier, P. A., & Davern, M. J. (2008). Measuring the effects of business intelligence systems: The relationship between business process and organizational performance. *International Journal of Accounting Information Systems*, 9(3), 135–153. <https://doi.org/10.1016/j.accinf.2008.03.001>
- Emam, A. Z. (2013). Critical success factors model for business intelligent over ERP cloud. *2013 International Conference on IT Convergence and Security, ICITCS 2013*, 13–16. <https://doi.org/10.1109/ICITCS.2013.6717819>
- Firebase. (2018). Retrieved March 5, 2018, from <https://firebase.google.com/>
- Ganesh, K., Sivakumar, S. M. (2014). *Enterprise Resource Planning : Fundamentals of Design and Implementation*. Springer International Publishing Switzerland. <https://doi.org/10.1007/978-3-642-31371-4>
- Gartner. (2015). Demand for Enterprise Mobile Apps Will Outstrip Available Development Capacity Five to One. Retrieved January 1, 2018, from <https://www.gartner.com/newsroom/id/3076817>
- Graeme Shanks, Peter B. Seddon, L. W. (2003). *Second-Wave Enterprise Resource Planning Systems: Implementing for Effectiveness*. *Second-Wave Enterprise Resource Management Systems* (Vol. 1). <https://doi.org/10.1017/CBO9781107415324.004>
- Griffith, C. (2017). *Mobile app development with Ionic 2 : cross-platform apps with Ionic, Angular, and Cordova*. Retrieved from https://books.google.co.uk/books?id=G6WkDgAAQBAJ&pg=PT10&lpg=PT10&dq=ionic+framework+first+release+date+2015&source=bl&ots=DH35CkMwn6&sig=jIIxbhb_A_FhpcEMRWij9NbXlqE&hl=en&sa=X&ved=0ahUKEwjlLjVo4rWAhXkbZoKHfvFBHwQ6AEIXTAJ
- Griffiths, J. (2016). Mastering Ionic 2.
- Hawking, P., & Sellitto, C. (2010). Business Intelligence (BI) Critical Success Factors. *ACIS 2010 Proceedings*, 11.
- He, Y., Zhang, D., & Fang, Y. (2017). Development of a mobile post-disaster management system using free and open source technologies. *International Journal of Disaster Risk Reduction*, 25(August), 101–110. <https://doi.org/10.1016/j.ijdr.2017.08.007>
- Hevner, A. R. (2007). Scandinavian Journal of Information Systems A Three Cycle View of Design Science Research A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems* © *Scandinavian Journal of Information Systems*, 19(192), 87–92. <https://doi.org/http://aisel.aisnet.org/sjis/vol19/iss2/4>
- Holzinger, A., & Slany, W. (2012). Multidisciplinary Research and Practice for Information Systems, 7465(March 2016). <https://doi.org/10.1007/978-3-642-32498-7>
- Huynh, M., & Truong, D. (2017). Hybrid App Approach: Could it mark the end of native app domination? *Issues in Informing Science + Information Technology*, 14, 49–65. <https://doi.org/10.28945/3723>
- Ionic. (2018). Ionic Framework. Retrieved January 27, 2018, from <https://ionicframework.com>
- jQuery Mobile. (2018). Retrieved February 23, 2018, from <https://jquerymobile.com/>
- Koupaei, M. N., Mohammadi, M., & Naderi, B. (2016). An Integrated Enterprise

- Resources Planning (ERP) Framework for Flexible Manufacturing Systems Using Business Intelligence (BI) Tools, 3(1), 1112–1125.
- Kudo, N., Yamauchi, T., & Austin, T. H. (2017). Access control for plugins in cordova-based hybrid applications. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, (2), 1063–1069. <https://doi.org/10.1109/AINA.2017.61>
- Latif, M., Lakhri, Y., Nfaoui, E. H., & Es-Sbai, N. (2016). Cross platform approach for mobile application development: A survey. *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, 1–5. <https://doi.org/10.1109/IT4OD.2016.7479278>
- Malavolta, I. (2016). Web-based Hybrid Mobile Apps : State of the Practice and Research Opportunities, 2016–2017. <https://doi.org/10.1145/2897073.2897133>
- Microsoft. (2018a). Visual Studio Code. Retrieved from <https://code.visualstudio.com/>
- Microsoft. (2018b). Visual Studio IDE. Retrieved from <https://visualstudio.microsoft.com/vs/>
- Mobile Operating System Market Share Worldwide | StatCounter Global Stats. (2018). Retrieved March 4, 2018, from <http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201401-201712>
- Mobile Solutions | Google Cloud Platform. (2018). Retrieved March 5, 2018, from <https://cloud.google.com/solutions/mobile/>
- Monk, E., & Wagner, B. (2009). *Concepts in enterprise resource planning*.
- Moroney, L. (2017). *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. The Definitive Guide to db4o*. <https://doi.org/10.1007/978-1-4302-0176-2>
- Newhook, R., Jaramillo, D., Temple, J. G., & Duke, K. J. (2015). Evolution of the Mobile Enterprise App: A Design Perspective. *Procedia Manufacturing*, 3(Ahfe), 2026–2033. <https://doi.org/10.1016/j.promfg.2015.07.250>
- NodeJS. (2018). Retrieved January 27, 2018, from <https://nodejs.org/en/>
- Novac, O. C., Marczin, R.-G., & NOVAC, M. C. (2016). Comparison of Hybrid Cross-Platform Mobile Applications with Native Cross-Platform Applications. *Journal of Computer Science & Control Systems*, 9(2), 24–27.
- Omar, K., & Gómez, J. M. (2016). A selection model of ERP system in mobile ERP design science research, Case study: mobile ERP usability. *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. <https://doi.org/10.1109/AICCSA.2016.7945791>
- Paulson, L. D. (2005). Web Applications with Ajax. *IEEE Computer*, 38(10), 14–17. <https://doi.org/10.1109/MC.2005.330>
- Perchat, J., Desertot, M., & Lecomte, S. (2013). Component based framework to create mobile cross-platform applications. *Procedia Computer Science*, 19, 1004–1011. <https://doi.org/10.1016/j.procs.2013.06.140>
- PHC. (2018). PHC Software. Retrieved January 1, 2018, from <https://www.phcsoftware.com/>
- Pordata. (2016a). Qual a percentagem de micros, pequenas e médias empresas no total de empresas. Retrieved January 1, 2018, from <https://www.pordata.pt/Portugal/Pequenas+e+médias+empresas+em+percentagem+do+total+de+empresas+total+e+por+dimensão-2859>
- Pordata. (2016b). Quantas PME existem. Retrieved January 1, 2018, from <https://www.pordata.pt/Portugal/Pequenas+e+médias+empresas+total+e+por+dimensão-2927>
- Portdata, I. and. (2018). Nº Subscribers of mobile land service in Portugal. Retrieved

- January 1, 2018, from
<https://www.pordata.pt/Portugal/Assinantes+++equipamentos+de+utilizadores+do+serviço+móvel-1180>
- Que, P., Guo, X., & Zhu, M. (2017). A Comprehensive Comparison between Hybrid and Native App Paradigms. *Proceedings - 2016 8th International Conference on Computational Intelligence and Communication Networks, CICN 2016*, 611–614. <https://doi.org/10.1109/CICN.2016.125>
- Ramos, M., Valente, M. T., & Terra, R. (2017). AngularJS Performance: A Survey Study, 1–13. Retrieved from <http://arxiv.org/abs/1705.02506>
- React - A JavaScript library for building user interfaces. (2018). Retrieved February 21, 2018, from <https://reactjs.org/>
- Robert Jacobs, F., & “Ted” Weston, F. C. (2007). Enterprise resource planning (ERP)-A brief history. *Journal of Operations Management*, 25(2), 357–363. <https://doi.org/10.1016/j.jom.2006.11.005>
- Shahzad, F. (2017). Modern and Responsive Mobile-enabled Web Applications. *Procedia Computer Science*, 110, 410–415. <https://doi.org/10.1016/j.procs.2017.06.105>
- Shahzad, F., Sheltami, T. R., Shakshuki, E. M., & Shaikh, O. (2016). A Review of Latest Web Tools and Libraries for State-of-the-art Visualization. *Procedia Computer Science*, 58(Euspn), 100–106. <https://doi.org/10.1016/j.procs.2016.09.017>
- Smutný, P. (2012). Mobile development tools and cross-platform solutions. *Proceedings of the 2012 13th International Carpathian Control Conference, ICC 2012*, (May 2012), 653–656. <https://doi.org/10.1109/CarpathianCC.2012.6228727>
- Statista. (2015). Most pressing long-running challenges facing software developers worldwide, as of 2015. Retrieved January 1, 2018, from <https://www.statista.com/statistics/627071/worldwide-developer-survey-pressing-developer-challenges/>
- Statista. (2018a). Number of apps available in leading app stores as of 1st quarter 2018. Retrieved September 1, 2018, from <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>
- Statista. (2018b). Number of mobile app downloads worldwide in 2017, 2018 and 2022 (in billions). Retrieved from <https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/>
- Tadger, R. (1998). Enterprise Resource Planning - ABI/INFORM Collection - ProQuest. Retrieved from <https://vpn2.iscte.pt/+CSCO+00756767633A2F2F66726E6570752E63656264687266672E70627A++/abicomplete/docview/226885690/fulltext/864A0B02892342DAPQ/1?accountid=38384>
- Tomišová, V., & Hudec, J. (2017). Modelling of the costs of decision support for small and medium-sized enterprises, 22–31. <https://doi.org/10.20470/jsi.v8i1.280>
- Torre, C. de la, & Calvert, S. (2016). *Microsoft Platform and Tools for Mobile Application Development*.
- Turban, E., & Volonino, L. (2011). *Information Technology for Management*. Jhon Wiley & Sons, Inc. (Vol. 8). <https://doi.org/10.1017/CBO9781107415324.004>
- Wajid, A., Junjun, P., Akbar, A., & Mughal, M. A. (2018). WebGraveStone: An online gravestone design system based on jQuery and MVC framework. *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development*,

- ICoMET 2018 - Proceedings, 2018-Janua*, 1–6.
<https://doi.org/10.1109/ICOMET.2018.8346322>
- Walker, J. D., & Chapra, S. C. (2014). A client-side web application for interactive environmental simulation modeling. *Environmental Modelling and Software*, 55, 49–60. <https://doi.org/10.1016/j.envsoft.2014.01.023>
- Wargo, J. M. (2012). *PhoneGap Essentials: Building Cross-platform Mobile Apps*. <https://doi.org/10.1007/s13398-014-0173-7.2>
- Xamarin. (2018). Mobile Application Development to Build Apps in C# - Xamarin. Retrieved January 27, 2018, from <https://www.xamarin.com/platform>