UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

A Framework for Creating Applications for Different Mobile Operating Systems

Pedro Jorge Madeira Tavares

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Pedro Ricardo Morais Inácio

Covilhã, outubro de 2014

I dedicate this dissertation to my parents for supporting me each step of the way.

Acknowledgements

This dissertation represents the end of a long academic journey. Even thought this section will not make justice regarding all persons that helped me along the way, it will emphasize those that most significantly contributed for its success.

First of all, I would like to thank my family for the support they provided me through my entire life and, in particular I must acknowledge my father António Jorge Tavares and my mother, Isaura Maria Madeira Tavares. Without their love, encouragement and assistance, I would have not finished this dissertation.

I would also like to express my gratitude to my supervisor, Professor Dr. Pedro Inácio, whose experience, knowledge and patience, contributed considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas, and his guidance through the various moments of my academic life. His skills both in and off the academic context have on occasion made me *GREEN* with envy.

I would like to acknowledge the Multimedia Signal Processing Covilhã Group (MSP-CV), of Instituto de Telecomunicações (IT), for providing the work environment in this developed masters programme.

Finally, a very special thanks goes out to my dear friend and partner in crime, Fábio Rodrigues, for taking time out from his busy schedule to help me to perfect this document, by reviewing it.

Thank you!

"É preciso viver, viver como homem comum entre homens comuns.
Só um homem comum pode fazer grandes coisas."

António Lobo Antunes (1942)

"It is never too late to be what you might have been."

George Eliot (1819-1880)

"People grow through experience if they meet life honestly and courageously.
This is how character is built."

Eleanor Roosevelt (1884-1962)

Resumo

O recente crescimento da quota de mercado de *smartphones* tem impulsionado a entrada de novas empresas no mercado das aplicações móveis. Essas empresas entendem, tipicamente, que estas peças de *software* devem ter ciclos de desenvolvimento e de vida mais curtos, a fim de melhorar a produtividade do trabalhador, e reagir às mudanças do mercado e necessidades dos clientes. Assim, é importante desenvolver aplicações móveis que executem em múltiplas plataformas simultaneamente com a mesma base de código, diminuindo o esforço, reduzindo os custos e obedecendo, sempre, às expectativas do utilizador final. Por estes motivos, muitas empresas têm concentrado algum esforço na procura ou desenvolvimento de novos meios para a rápida e descomplicada conceção de aplicações móveis.

Nesta dissertação é relatado um trabalho de pesquisa sobre sistemas para desenvolvimento de aplicações móveis e *web* multi-plataforma, que concretiza o primeiro objetivo deste mestrado. Neste contexto, foram analisadas várias ferramentas existentes, descrevendo-se as suas vantagens, desvantagens e a sua importância na construção de aplicações móveis. Estas aplicações podem ser classificadas em diferentes tipos, nomeadamente em *nativas*, *híbridas* e *não nativas*. Estas classes são igualmente objeto de estudo. Este estudo permitiu compreender qual a ferramenta mais popular na indústria de desenvolvimento de *software* para dispositivos móveis. Segundo algumas pesquisas produzidas na área, a PhoneGap é a ferramenta que oferece a mais notável gama de funcionalidades e a Ferramenta Multi-Plataforma (da designação inglesa *Cross-Platform Tool* (CPT)) que provê sustento à grande maioria dos sistemas para Desenvolvimento Móvel Multi-Plataforma (da designação inglesa *Cross-Platforms to Mobile Development* (CPMDs)). Estudaram-se, de seguida, este tipo de sistemas (CPMDs), que são plataformas baseadas na *web*, e que usam as CPTs como base para viabilizar uma comunicação entre a aplicação final não nativa e a Interface de Programação de Aplicações (do inglês *Application Programming Interface* (API)) nativa do sistema operativo móvel. De forma a compreender o seu funcionamento e os benefícios do seu uso, foi feita uma análise a um conjunto de sistemas semelhantes.

O segundo objetivo principal deste programa de mestrado compreendia a implementação de uma CPMD. Com base no conhecimento adquirido na fase inicial, foi efetuado o levantamento de requisitos para a sua conceção e elaborada a engenharia de *software*. Durante o desenvolvimento da CPMD foram produzidas algumas otimizações de código e implementados, também, alguns mecanismos de segurança modernos de forma a fornecer garantias de disponibilidade, confidencialidade e integridade. Como este sistema entrega aplicações móveis nativas e não nativas, foi elaborado um estudo comparativo entre os dois tipos de aplicações, que permitiu compreender quais as suas diferenças tanto a nível de performance, como de tempos de execução e memória. Este estudo concretiza outras das contribuições. Para além da dissertação, o resultado mais visível do trabalho realizado é um protótipo completamente funcional de uma

CPT e uma CPMD.

Finalmente, e para concluir o trabalho aqui descrito, foi produzido e distribuído um questionário a um grupo de pessoas, a quem foi pedido que usassem o protótipo desenvolvido. A motivação para a sua concretização foi compreender se as aplicações móveis, bem como as CPMDs, começam já a ser reconhecidas como ferramentas de negócio importantes nos dias de hoje e também se as aplicações entregues pelo protótipo desenvolvido vão de encontro às necessidades dos inquiridos. O retoque de alguns detalhes de implementação, *design* e até na usabilidade do sistema, seria o foco central de uma próxima fase de desenvolvimento.

Palavras-chave

Aplicações Móveis, Aplicações Móveis Nativas e Não Nativas, Programação Móvel, Ferramentas Multi-Plataforma, Segurança em Plataformas Baseadas na *Web*, Sistemas para Desenvolvimento de Aplicações Móveis Multi-Plataforma.

Resumo alargado

Introdução

Este capítulo escrito em língua Portuguesa vem resumir, de modo mais alargado que o resumo, o corpo desta dissertação. Começa por apontar o enquadramento da mesma, fluindo a discussão para a descrição do problema e enumeração das principais contribuições. São seguidamente descritos conceitos relevantes para a boa compreensão do que resta do capítulo, juntamente com uma breve discussão de trabalhos relacionados com esta dissertação. A engenharia de *software* e o método adotado para conceber um protótipo são descritos nas seguintes secções. Na antepenúltima secção são apresentados os testes do estudo comparativo entre aplicações nativas e não nativas e os resultados obtidos destes, pelo que são incluídos alguns comentários tendo em conta o problema. A penúltima secção apresenta os resultados de um questionário, com o objetivo de apurar se o protótipo concebido é capaz de satisfazer o público-alvo. O último capítulo apresenta as principais conclusões obtidas, bem como algumas direções para trabalho futuro.

Nos capítulos escritos em língua Portuguesa, os acrónimos que são utilizados aparecem em Português nas suas formas longas, no entanto as suas formas curtas são mantidas com o respetivo acrónimo em língua Inglesa, para efeitos de consistência. Não obstante, os acrónimos cuja tradução não é diretamente possível, por se referirem a serviços ou aplicações, aparecem em itálico como estrangeirismos.

Enquadramento, Descriçao do Problema e Contribuições Principais

Atualmente, a programação móvel e as aplicações móveis são dois tópicos sobejamente debatidos no mercado de *software*. Como prova desse facto, grande parte da população possuí um dispositivo móvel, pois este tem um forte impacto nas suas vidas afirmando-se, decididamente, no seu quotidiano, quer seja para consultar o seu *e-mail* pessoal ou navegar na *Internet*. Ao longo dos anos, tem sido notório o vasto crescimento e desenvolvimento do conjunto de ferramentas e soluções móveis, nesse sentido é possível concluir que as empresas apresentam interesse em acompanhar esta nova tendência. Este facto é um claro sinal que o mercado de aplicações móveis é um mercado com futuro. A eficiência no desenvolvimento de soluções, o próprio custo e esforço na conceção de aplicações móveis são fatores de sucesso deste mercado. A redução de custos, tanto computacionais como humanos, é alcançado por força de ferramentas que visam automatizar todo este processo. As Ferramentas Multi-Plataforma (da designação inglesa *Cross-Platform Tools* (CPTs)) e as Multi-Plataformas para Desenvolvimento Móvel (do inglês *Cross-Platforms to Mobile Development* (CPMDs)) são ferramentas que possibilitam esse cómodo processo de desenvolvimento, reduzindo o número de entidades envolvidas na edificação de uma nova aplicação e até o problema das versões, impostas pelos sistemas op-

erativos móveis. Estas ferramentas acompanham a tendência das páginas *web* e podem ser desenvolvidas por pessoal especializado na área *web*. Usando JavaScript é possível que qualquer aplicação móvel comunique com a Interface de Programação de Aplicações (do inglês *Application Programming Interface* (API)) nativa do sistema operativo móvel. O advento destas metodologias e técnicas autoafirma o uso de ferramentas e plataformas centralizadas para a criação de aplicações móveis. Assim sendo, esta dissertação de mestrado propõe apresentar um trabalho relacionado com a área, descrevendo algumas das metodologias empregues na conceção de aplicações móveis. Como já mencionado, as CPTs e as CPMDs são o objeto de estudo deste programa.

Para concluir o que foi dito, este capítulo também apresenta as contribuições deste trabalho de mestrado, que podem ser definidas como se segue:

- Fornece um estudo detalhado e uma comparação de plataformas que estão atualmente disponíveis para a construção de aplicações móveis.

- Disponibiliza a documentação para cada uma das plataformas analisadas, e são descritas todas as boas práticas para a construção deste tipo de *software*.

- Apresenta a engenharia de *software* e implementação de um protótipo completamente funcional, tendo em conta a sua usabilidade e aspetos de performance e segurança.

- É apresentado um estudo comparativo da performance de aplicações nativas contra aplicações não nativas Android. Deve ser tomada como nota, que o desenvolvimento baseado nestas plataformas é capaz de entregar soluções para os diferentes sistemas operativos móveis, nomeadamente Android e FirefoxOS. Algumas das funcionalidades são convertidas, pela plataforma, para código nativo, enquanto outras, são embebidas em *web-views*.

Trabalho Relacionado

O trabalho descrito nesta dissertação pressupôs um estudo prévio das ferramentas adotadas pelos programadores na conceção de aplicações móveis e ainda, uma avaliação às suas limitações, vantagens, tendo em conta o seu contexto de utilização. Este estudo é apresentado no capítulo 2. De uma forma resumida, sabe-se que a procura de aplicações móveis tem aumentado radicalmente, logo, o advento de novas tendências e exigências na conceção de aplicações móveis auferiu um grande destaque e visibilidade. São, portanto, necessárias metodologias mais ágeis e que tenham como foco a rápida e descomplicada criação de aplicações. As CPTs, por sua vez, impulsionaram esta nova realidade. Com base em cliques, com o auxílio de tecnologias orientadas à *web*, isto é, Linguagem de Marcação de Hipertexto (abreviação para a expressão inglesa *HyperText Markup Language* (HTML)), *Cascading Style Sheets* (CSS) e JavaScript, tornou-se simples e rápido desenvolver aplicações móveis não nativas. Estas ferramentas, na sua maioria,

usam uma *web-view* para embeber todo o conteúdo oferecido, nomeadamente texto, imagens, ou outras funcionalidades disponibilizadas pela plataforma, como *streaming*, *plugins* que permitem interação com as redes sociais, etc. Elas também proporcionam a interação com componentes nativos do sistema operativo móvel. Essa intercomunicação é alcançada através de chamadas remotas, provocadas por código JavaScript. O tipo de aplicações entregues por este tipo de abordagem são chamadas de aplicações híbridas (ver figura 2.2), onde é estabelecida uma ponte (*bridge*, do Inglês) entre a API nativa e a *web-view* que contém todo o conteúdo produzido.

Atualmente existem algumas ferramentas desta linha. De forma a compreender o seu destaque na indústria bem como as suas valências e limitações, estas foram objeto de um trabalho de pesquisa. Após a concretização dessa pesquisa, percebeu-se que o CPT PhoneGap é aquela que oferece uma melhor experiência aos programadores. Esta detém um específico leque de funcionalidades, uma documentação por excelência e faculta uma aprazível ambientação aos seus utilizadores.

Como a segunda parte deste mestrado foca a construção de uma CPMD, isto é, uma plataforma *web* que possibilita a construção e entrega de aplicações móveis, foi fundamental a pesquisa elaborada ao longo deste programa. Sendo o PhoneGap a CPT de eleição na ótica de muitos desenvolvedores, não é de estranhar que seja a ferramenta adotada pela grande parte destas plataformas *web*. Esta consente que as aplicações por elas produzidas consigam comunicar com o sistema operativo móvel. A interface oferecida por este conjunto de plataformas possuí como base uma página *web*, que através de *drag-and-drop* ou *drop-down* menus, permite o desenvolvimento de aplicações. Uma das suas grandes valências, é sem dúvida, a camada de abstração oferecida, pois o programador poderá delinear um protótipo, e esse ser concebido para os mais diversos sistemas operativos móveis, sem sequer, ter de pensar nas diversas APIs nativas, como por exemplo, do Android ou iOS. Como é um tema ainda recente, a maioria destas plataformas são na sua maioria pagas, permitindo unicamente aos novos utilizadores um pequeno período de experimentação, que vai de 15 a 30 dias. A melhor forma de produzir uma análise de requisitos e a construção da engenharia de *software* para o protótipo proposto, foi o estudo comparativo entre as CPMDs desenvolvido na secção 2.6, onde também foram produzidas algumas tabelas com detalhes pertinentes para os restantes capítulos deste programa.

Engenharia de Software

Tendo em conta o estudo comparativo que terminou o capítulo 2 desta dissertação, foi exequível esquematizar a engenharia de *software* para o protótipo desenvolvido. Inicialmente, foi imprescindível um levantamento de requisitos, de forma a planear a melhor abordagem e oferta de funcionalidades ao público-alvo. Todo o trabalho desenvolvido ao longo do capítulo 2, serviu como suporte na definição e concretização da engenharia de *software* do sistema. Ao longo

deste capítulo, foram identificados os seus atores e casos de uso. O auxílio dos diagramas de Casos de Uso oferecidos pela Linguagem Unificada de Modelagem (da designação inglesa (*Unified Modeling Language* (UML)) proporcionaram estruturar e apresentar, na forma de esquema, três diagramas. O diagrama de caso de uso geral, presente na figura 3.1, esquematiza as dependências entre atores e casos de uso. É possível também constatar, que o administrador é responsável por gerir todos os utilizadores do sistema. Por outro lado, os utilizadores do sistema têm disponível todo um conjunto de funcionalidades ao nível da gestão e desenvolvimento dos seus projetos/aplicações.

Também ao longo deste capítulo é possível observar o diagrama de classes do sistema (ver figura 3.5), os diagramas de atividades, que representam os comportamentos de alguns casos de uso mencionados e o diagrama de componentes (ver figura 3.4) que apresenta a esquematização do modelo físico do sistema. Por fim, é apresentado o diagrama entidade-relacionamento da base de dados, que oferece sustento a todo o sistema (figura 3.6).

Implementaçao e Segurança da Plataforma

O desafio que se colocava na etapa a que esta secção diz respeito era a conceção de uma CPMD. Aqui, são descritos os detalhes que enfatizam a implementação do protótipo, aqueles que se concluíram ser mais relevantes e alvo de alguma discussão. Esta plataforma carateriza-se pela possibilidade de entrega de aplicações capazes. Também viabiliza o desenvolvimento de aplicações móveis para Android, para FirefoxOS e para a *web*, nomeadamente HTML5. O protótipo assenta num modelo cliente-servidor, e o servidor detém a responsabilidade de atender todos os pedidos solicitados pelos utilizadores. Ainda neste capítulo, é possível perceber através dos esquemas aduzidos pela figura 4.1 e figura 4.2, a organização interna do sistema e de uma diretoria do utilizador. Nesta última, são armazenados todos os ficheiros e dependências relativas a cada projeto/aplicação desenvolvida. No que diz respeito aos projetos, e como já mencionado no capítulo 3, a análise de requisitos e o trabalho relacionado efetuado ao longo deste programa contribuíram, em muito, para a seleção de funcionalidades a implementar. Estas funcionalidades podem ser observadas a partir da tabela 4.1, onde são apresentadas 18 das 49 funcionalidades que compõem o sistema. É possível encontrar um vasto leque de possibilidades, para distintos contextos, nomeadamente funcionalidades direcionadas para negócios, para restauração, funcionalidades baseadas nas redes sociais, processamento de imagem, etc. Para enaltecer o trabalho desenvolvido ao longo deste programa, foram apontados alguns detalhes de segurança, que foram implementados durante a conceção do protótipo. Foram também concretizadas algumas otimizações a nível de performance com o foco na qualidade do serviço. Este capítulo termina com a análise estatística e discussão sobre a performance do protótipo aqui desenvolvido.

Implementação Nativa e Não Nativa

Tendo em conta o último ponto dos objetivos desta dissertação, descrito na secção que inicia este capítulo, ambas as versões, nativa e não nativa, foram sujeitas a um conjunto de testes de forma a avaliar a sua performance, tempos de execução e espaço ocupado em memória. Esta matéria não tem sido alvo de muitas pesquisas, apesar do termo *mobile* se ter enraizado no quotidiano de cada indivíduo. Porém, subsiste algum esforço e investigação nesta área, mas ainda sem muitos resultados em concreto.

A fim de melhor quantificar a diferença em termos de desempenho e requisitos do sistema entre aplicações nativas e não-nativas, que era também um objectivo neste mestrado, foi realizado um estudo comparativo.

Para a concretização do estudo foram, portanto, usados dois *smartphones* distintos, tanto a nível de *hardware* como de *software*. As aplicações produzidas foram selecionadas à priori, com o objetivo de constituírem um estudo firme e bastante discriminativo. Como observado a partir da tabela 5.3, foram aplicadas 10 recolhas para cada objeto de estudo, em cada uma das versões (nativa e não nativa) da aplicação, nos dois *smartphones*. No final, constatou-se que as aplicações não nativas possuem um desempenho inferior comparativamente às aplicações nativas. Assim, e com o natural avanço e melhoria da tecnologia, o processamento e desempenho dos próprios dispositivos móveis proporciona aos utilizadores um melhor conforto. Desta forma, a performance aplicacional é posta em segundo plano. Se o foco da aplicação não nativa, por exemplo, for um estreito aglomerado de pessoas, então o uso de plataformas cruzadas para o seu desenvolvimento é, sem dúvida, a melhor opção. Esta permite um desenvolvimento por excelência, veloz e para um vasto número de sistemas operativos. Em caso contrário, se o público-alvo for um amplo número de utilizadores (na casa das centenas ou até milhares), o uso de programação nativa para a conceção de uma aplicação robusta, e com um nível de performance e otimização perfeitamente satisfatório, será, decerto, a melhor abordagem a seguir.

Testes e Feedback do Utilizador

Finalmente, este capítulo apresenta os resultados de um inquérito, que visa esclarecer se a aplicação concebida ao longo deste trabalho de mestrado era capaz de satisfazer e responder às necessidades do utilizador final. Para alcançar este último objetivo, foi configurada a rede local (*home network*) que oferecia comunicação com a *Internet* ao servidor, onde foi necessário configurar o *router* com a regra *port forwarding*, para que o tráfego fosse automaticamente direcionado para o servidor (ver figura 6.1). Para recolher o *feedback* dos utilizadores foi elaborado um questionário, que foi entregue através do *Google Forms*. Este questionário continha respostas de escolha múltipla e foi distribuído, em língua portuguesa, a um diverso grupo de utilizadores, 39 inquiridos, 13 pessoas do sexo feminino e 26 do sexo masculino. Após a análise

dos gráficos presentes neste capítulo, assim como os restantes no apendice D, foi possível concluir que a plataforma obedeceu às expectativas impostas pela grande parte dos utilizadores. Similarmente, foi possível perceber que alguns utilizadores partilham da opinião que os *smartphones*, e os dispositivos móveis em geral, são um objeto insubstituível e fundamental nas suas vidas. No entanto, foi-lhes questionado se estariam dispostos a pagar um tipo de licença para usufruir de um destes sistemas. A resposta com maior abundância foi `não`, no entanto, esta resposta poderá estar relacionada com uma outra questão colocada logo no início do questionário, que abordou o utilizador no sentido de averiguar se detinha conhecimentos sobre este tipo de plataformas, onde novamente, a resposta `não` foi aquela com superior tendência.

Conclusões e Trabalho Futuro

As conclusões deste trabalho são enumeradas no capítulo 7. O desenvolvimento de aplicações móveis para os inúmeros sistemas operativos móveis é uma tarefa árdua. Derivado a esta situação surgiram as primeiras CPTs e CPMDs. Estas foram objeto de análise no capítulo 2 desta dissertação de mestrado. O foco deste capítulo foi direcionado para o estudo destes dois géneros de ferramentas, indicando portanto, as suas vantagens e desvantagens. Como a parte prática desta dissertação recaía na construção de uma completa CPT e CPMD, foi necessário estudá-las meticulosamente. Esta ferramenta, dois em um, foi aprovada a nível de segurança e performance. Relativamente a esta última, foram motivadas algumas melhorias no seu desempenho, através de alguns acertos mencionados no capítulo 4. Por fim, através de um questionário referido no último capítulo desta dissertação, foi possível captar algum *feedback* dos utilizadores relativamente à plataforma elaborada. No entanto, foram deixadas algumas linhas, provenientes das observações deixadas pelos utilizadores, para o trabalho futuro.

Uma das linhas mais diretas de trabalho futuro resultantes deste mestrado diz respeito à melhoria da estrutura desenvolvida. A pesquisa discutida no capítulo 6 enfatizou que ainda há muito espaço para melhoria em termos de interface do usuário, funcionalidades e fluxo de trabalho. Alguns dos participantes deixaram notas com boas sugestões sobre todos os tópicos acima mencionados. Em termos de melhoria da Expertnear Cross-Platform to Mobile Development (CPMD), o mais interessante para ser adicionado a curto prazo é o suporte para publicar o aplicativo móvel resultante, nas respectivas lojas de software nomeadamente o Google Play, de modo a automatizar todo o processo de desenvolvimento. Isso deixaria a *framework* numa posição melhor para competir com as soluções existentes. A idealização e criação de uma aplicação com a finalidade de estudar as CPMDs mencionadas, onde fosse examinada a usabilidade de cada plataforma e a ajuda concedida a cada passo de desenvolvimento, poderia ser benéfico no encalço de outros resultados sobre o tema. Finalmente, algumas melhorias a nível de *design* e usabilidade seriam tidas em conta num trabalho futuro.

Abstract

The recent growth of the market share of smartphones has been bringing new companies into the mobile applications market. For these companies, these software pieces should typically have smaller life and development cycles in order to improve the productivity of the workers and react to market changes and to the clients expectations. To reduce the effort required for development and improve cost efficiency, it is thus important to develop mobile applications that are able to run on several platforms simultaneously with same code base, while assuring that the expectations of the final user are fulfilled. Because of this, many companies have been focusing some effort on searching or developing uncomplicated new means for rapidly conceiving and delivering mobile applications.

This dissertation discusses a research work about systems for developing cross-platform web and mobile applications, which was the first main objective of this masters. Within this context, several existing tools were analysed, and their advantages, disadvantages and importance in the construction of mobile applications were described. These applications may be classified as native, non-native and hybrid. These classes were also subject to analysis. This initial study enabled understanding which was the most popular tool in the industry of software development for mobile devices. According to some reports on the area, PhoneGap is the one offering the most notorious panoply of functionalities and the Cross-Platform Tool (CPT) feeding most of the Cross-Platform to Mobile Development (CPMD) systems. CPMDs, which are web based platforms that elaborate on CPTs to provide communication between an application and the native Application Programming Interface (API) of a given mobile Operating System (OS), were then studied. In order to understand their way of functioning and benefits, an analysis to several similar systems was performed.

The second main objective of this masters programme was to implement a CPMD. Starting from what was learned in the initial phase of the work, a requirement analysis for the conception of such systems was performed and, subsequently, the software engineering was elaborated. Some code optimizations of the code were produced during the development of the CPMD, in which several state-of-the-art security mechanisms were also implemented to assure availability, confidentiality and integrity. Since this system delivers native and non native mobile application, a comparative study between both types of applications was also performed, which enabled to better understand the trade-offs and differences in terms of performance, execution times and memory. This study embodies another contribution of this work. Apart from the dissertation, the completely functional prototype of a CPMD integrated with a CPT comprises the most visible outcome of this masters.

Finally, concluding the work described herein, a survey was elaborated and delivered to a group

of persons, who were asked to use the developed prototype. The idea was to assess if mobile applications and platforms like CPMDs were already seen as important business tools nowadays and also if the applications delivered by the prototype meet the respondents expectations. The improvement of some implementation, design and usability details was pointed out a possible future line of work.

Keywords

CPMD, CPT, Cross-Platform Mobile Development, Cross-Platform Tools, Mobile Applications, Mobile Programming, Native and Non Native Mobile Applications, Security in Web-Based Platforms.

Contents

List of Figures

List of Tables

Acronyms

ACM Association for Computing Machinery

API Application Programming Interface

APK Android Package

BaaS Backend as a Service

CDN Content Delivery Network

CSS Cascading Style Sheets

CPIDE Cross-Platform Integrated Development Environments

CPMD Cross-Platform to Mobile Development

CPT Cross-Platform Tool

DBMS Data Base Management System

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

Info Sec Information Security

IP Internet Protocol

IT Instituto de Telecomunicações

JDK Java Development Kit

JIT Just In Time

LAN Local Area Network

MAD Mobile Application Development

MSP-CV Multimedia Signal Processing Covilhã Group

OS Operating System

OWASP Open Web Application Security Project

PBKDF2 Password-Based Key Derivation Function 2

PHP Hypertext Preprocessor

QML Qt Modeling Language

QoS Quality of Service

QR Quick Response

RAM Random-access Memory

RIA Rich Internet Application

RSS Really Simple Syndication

SDK Software Development Kit

SHA Secure Hash Algorithm

SMS Short Message Service

SQLi SQL Injection

SSL Transport Layer Security

SVG Scalable Vector Graphics

TV Television

TCP Transmission Control Protocol

UI User Interface

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

XSS Cross-site Scripting

xxx

Abbreviations

Please consider the following abbreviations with the respective meaning when later invoked in the text:

e.g.	originates from the Latin expression *exempli gratia* which means "for example".
i.e.	originates form the Latin expression *id est* which means "that is" or "in other words".

Chapter 1

Introduction

This dissertation elaborates on the subject of mobile cross-platforms, as well as on the steps required to implement one of these systems. The next section presents the motivation and scope behind this work, the next-to-last section presents the main contributions to build a useful and secure web framework to develop mobile applications and the last section describes the contents of each chapter of this dissertation.

1.1 Motivation and Scope

Developing mobile applications comprises a challenging task, where technology and creativity are essential. Nowadays, most citizens of developed countries posses a mobile device, considered by many as an indispensable object in the life of everyone. A mobile device, like a smartphone, is a means to assure the communication requirements between individuals, groups, companies and services, representing a mark of the human genius. Mobile devices and the Internet are simultaneously the results and key pieces of globalization. Their history can be traced up to 1914, when Heinrich Hertz transmitted the first codes wirelessly between continents [int].

The market of mobile devices has been promising and fast-growing for more than a decade now. Gartner states that 967,775.8 million smartphones were sold in 2013 [Gar14]. The opportunities supported by these devices and associated technology imply changes to the way software is built and released and have a direct impact on the productivity and leisure time of people. In a 2010 survey by Forrester Research, 75% of companies defended that the productivity of their employees was increased through the deployment of mobile applications, enabling faster decision making and problem resolution, better customer experience and other benefits [For10]. The lifecycles of mobile applications are also becoming fundamentally different from software for desktop computers, both in terms of the development and production phases. These changes are reactions to customer requirements and market volatility. Some mobile applications are developed in a few days, requiring effective tools for faster delivery and maintenance of software releases [Gar09].

Currently, an increasing demand for many types of business applications can be observed on the market of mobile applications [Gar09], including applications that primarily target consumers. Affordable smartphones such as the Samsung Galaxy Fresh, which started with a price of around €120 in Portugal, might make such devices even more accessible for end users. Online markets

for mobile applications are growing as well. Through a press release, Apple reported that, in 2013, the Application Store made no less than $10 billion with sales of applications and games [App14]. Applications of popular online platforms such as Facebook or YouTube, but also applications with a very specific purpose, such as Amazon Mobile, Dropbox[1] or RunKeeper[2], have been installed hundreds of millions of times from the stores.

All these trends and facts lead to the conclusion that the mobile applications market is a market of the future, and is already a great opportunity for businesses. While the primary goal for a business is of course to improve profit, it is necessary to figure out how this goal can be achieved in terms of efficient development, cost-effective delivery and customer adoption. Another factor of success is to keep development costs low. They can be cut down by using fewer resources (e.g., hardware), having cheaper or free software licenses, using efficient tools (e.g., for development and release automation, or development environments) and reducing expenses for personnel. Maintenance costs, such as expenses for bug fixing and (professional) support, also have to be considered. In order to combat what was said, and even the problem of versioning of mobile Operating System (OS), tools for development of cross platform mobile applications have emerged. These tools eventually dictate an evolution similar to the one that happened to web pages, sites and applications some years ago. While the development of web content and applications could only be initially performed by people with expertise on the area, nowadays, several frameworks and tools support their design, development and maintenance via *what you see is what you get* extremely intuitive user interfaces. It interesting to notice that, actually, some web related technologies are emerging as suitable alternatives for creating mobile applications that run on several platforms, with JavaScript and HyperText Markup Language (HTML)5 assuming key roles on that story.

This dissertation is precisely focused on the subject of the development of cross platform mobile applications, motivated above. This subject is already vast and it is evolving quickly, thus comprising a hot topic nowadays.

Under the 2012 version of the Association for Computing Machinery (ACM) Computing Classification System, a *de facto* standard for computer science, the scope of the masters programme, reflected in this dissertation, falls within the categories named:

- **Security and privacy** ∼ **Software security engineering**;
- **Security and privacy** ∼ **Web application security**;
- *Security and privacy* ∼ *Mobile platform security*;

[1] Synchronization and sharing of files between multiple devices
[2] Tracking and timing of sport workouts like running

- *Security and privacy* $\sim$ *Database and storage security*;
- *Networks* $\sim$ *Network services*;
- *Software and its engineering* $\sim$ *Software creation and management*.

1.2 Problem Statement and Objectives

The problem addressed in this masters degree programme is the one of providing a means to securely and easily develop applications for mobile devices. Concretely, the problem is that of creating a framework, supported by web technologies, that enables a user to build such applications via a graphical and intuitive user interface only. It must not be necessary for the typicall user of the framework to be an expert in order to produce cross-platform mobile applications.

This work is thus defined by three main objectives: (i) study existing technologies, tools and frameworks addressing the problem at hands or that may help in solving it; (ii) implement the prototype of a system to develop mobile and web applications that run on multiple platforms; and (iii) discuss and quantify the trade-offs between native and non-native mobile applications, potentially via empirical experiments. Most of the existing tools and frameworks for developing cross platform mobile applications, also known as Cross-Platforms to Mobile Development (CPMDs), are paid with a limited trial period. A secondary objective of this masters is thus to deliver an open source and free CPMD and describe the requirements and details that can be used to build such a system. The CPMD should support exporting applications to platforms, such as the web and Android simultaneously, and should be tested in a real deployment scenario. To develop a secure, efficient and scalable system comprise also important objectives of this work.

1.3 Adopted Approach for Solving the Problem

In order to achieve the objectives described in the previous section, this masters programme was divided into the following phases:

1. Contextualization with the problem addressed in the dissertation and with related research work on this area of knowledge. This part included an overview of press and industry reports regarding mobile application development, and also studying the tools available in the market, mostly to identify the pros and cons of the most important Cross-Platform Tools (CPTs) and CPMDs. Additionally, it included preparing a comparative study of some CPMDs and their benefits;

2. Requirement analysis and complete software engineering of a CPMD. This phase comprised also the planning of the security mechanisms to be integrated in the system;

3. Implementing a capable web framework (CPMD) with basis on the outputs of the previous phases;

4. Improvement of the developed system, mostly by introducing several performance optimizations;

5. Perform a comparative study regarding the performance of native and non-native applications;

6. Finally, in order to obtain feedback from potential users of the developed system, a survey was planned and distributed, and the results were scrutinized.

The overall organization of the dissertation (refer to section 1.5) reflects the way this work was structured.

1.4 Main Contributions

The main contributions of this masters programme and dissertation are as follows:

1. It delivers a detailed study and comparison of the frameworks that are currently available to construct mobile applications;

2. It elaborates on the engineering and implementation of one of those frameworks, while taking usability and, most importantly, security, into consideration, delivering also a functional prototype;

3. It provides the documentation for such framework, and describes the required security mechanisms and best practices for this type of software;

4. It presents a comparative evaluation of the performance of native against non-native applications in the Android OS. It should be noted that the developed web based framework is able to produce packages to different mobile OSs, namely Android and FirefoxOS. Some of the features are translated into native by the framework, while others are embedded in a web views.

1.5 Dissertation Overview

The main body of the dissertation is composed by seven chapters. The contents of each one of the chapters can be summarized as follows:

- Chapter 1 -- **Introduction** -- presents the scope and the motivation behind the work described in this dissertation, as well as the addressed problem and objectives. It includes a

subsection describing its main contribution for the advance of knowledge and the approach taken to fulfill the objectives. The dissertation structure is also briefly discussed at the end of this chapter.

- Chapter 2 -- **Related Work** - contains a discussion on the methodologies and tools used to create mobile applications, along with their advantages and disadvantages. It also contains a study on some commercially or freely available CPMDs.

- Chapter 3 -- **Software Engineering** -- is dedicated to the software engineering of the framework developed along this masters programme. It contains the requirements analysis, the identification of Use Cases, and the discussion of some activities and class diagrams. It includes also the discussion for the Components Diagram and for the modeling of the framework supporting database.

- Chapter 4 -- **Implementation and Security of the Framework** -- describes all the methods utilized to build a capable and efficient solution. Additionally, the most interesting findings of this initial phase are discussed, along with the means used to implement security features.

- Chapter 5 -- **Native and Non-Native Implementation** -- discusses specific details related with these two ways of implementing applications, along with their advantages and disadvantages. It elaborates on an experiment regarding the performance, time of response and memory used by native and non-native mobile applications, so as to shed a concrete light into this subject.

- Chapter 6 -- **Tests and User Feedback** -- starts by presenting the user tests to the developed prototypes and evolves to the discussion of the results of that evaluation.

- Chapter 7 -- **Conclusions and Future Work** -- summarizes the main conclusions of this masters programme, with focus on the results, and identifies some future research directions.

For the sake of clarity, some of the material produced along this masters were included as annexes. The contents of these annexes can be briefly described as follows:

- Appendix A -- **Advantages and Disadvantages of CPTs** -- contains all the tables and an appropriate discussion concerning the CPTs discussed in chapter 2.

- Appendix B -- **Scenarios of Use Cases** -- includes all the scenarios for the use cases presented in chapter 3.

- Appendix C -- **Dataset: Native and Non-Native Implementation** -- contains all the tables

and charts produced within the scope of the study discussed in chapter 5.

- Appendix D -- **Results of the Questionnaire** -- contains all the charts produced within the scope of the survey discussed in chapter 6.

Chapter 2

Related Work

This chapter discusses the paradigm of CPTs, presenting the similarities between the several available solutions, their advantages and disadvantages. It also presents the general architecture of such platforms. An analysis of the several characteristics of web frameworks to mobile development, currently available in the market, is also included herein, in the form of a table. Notice that, during the remaining part of this dissertation, these frameworks for mobile development will be generally referred to as CPMDs.

The chapter is structured as follows. After a brief introduction to the subject at hands, section 2.2 discusses the subject of native mobile application development. Section 2.3 is devoted to the identification of the several types of cross-platform tools while the general architecture of CPTs is described in section 2.5. Towards the end of this chapters, a comparative study of the several currently available CPMDs is delivered in section 2.6.

2.1 Introduction

These days, Mobile Application Development (MAD) is quite challenging, even in terms of choice of Integrated Development Environment (IDE) and technology, since there are several platforms and toolkits for development. As technology advances, mobile devices are becoming more powerful in terms of storage, processing and communication capabilities. With less costly technology, they are more accessible to consumers. The landscape of mobile platforms has suffered a major evolution in the recent past. While BlackBerry [Bla14], Bada [Sam12] and Symbian [Sym12] failed to reach out to the masses, iOS [Appa] and Android [Goob] have won the war of mobile platforms. More recently, Firefox has made also a release of a mobile OS, known as the Firefox OS [Fir]. In the era of smartphones and tablets, mobile applications provide added value to enterprise services and several industries including transport, e-commerce, net banking, travel and retail. Thus, it is of critical importance for a mobile application provider to attract more developers in order to boost external investment and revenue. Lately, mobile developers are confronted by a duopoly. Google (with Android) and Apple (with iOS) devices account for more than 90% of the smartphone market [Bri14], but there is no clear winner in terms of market share. Currently, the corporations have the ability to create two separate applications, but are hesitant to do so, because of the amount of work required to support and maintain two applications over the life cycle of a product. Fortunately, writing one application for iOS and another for Android is not the only option. In order to automate the process of MAD, some companies are

designing and developing toolkits or CPTs, such as PhoneGap [Phob], Titanium [Tit], etc., and also CPMDs, like MobileRoadie [Bro], TheAppBuilder [The12], GoodBarber [Gooa] and others.

Typically, each mobile platform, and its respective Software Development Kits (SDKs), represent unique challenges to developers and companies. This normally contributes to increasing the time of development and the cost to market factor. In order to contradict this, the developer must have thorough knowledge of the platforms. This is the point where CPMDs come in. Most of them make use of toolkits currently available in the development market, as for example the PhoneGap and Titanium. Development platforms are built over these tools, using HTML, Cascading Style Sheets (CSS) and JavaScript as the basis to generate the interface. Meanwhile, the applications can be generated for various mobile OSs, such as iOS, Android, Windows Phone [Mic] and BlackBerry. This process is helpful only when the importance given to the application cross-platform feature is critical, i.e., when one wants to maximize audience reach. Using this platforms, the cost of development and time to market are reduced. Fortunately, the number of CPMDs and CPTs for mobile devices is growing in terms of quantity and quality.

2.2 Native Mobile Applications Development

Before presenting the analysis of CPTs and CPMDs, a brief introduction to native implementation is pertinent. This topic will, nonetheless, be further discussed in chapter 5 of this dissertation.

Native application development refers to implement software specially designed for a specific platform, hardware, processor and its set of specific instructions. The advantages of native application development are predictable performance, streamlined support, native User Interface (UI), native Application Programming Interface (API) and coherent library updates. Native development requires a specific SDK and programming language. This comes usually with a different development cost regarding the different software and platform used [Dan10]. These applications are executable binary files of an application that will be installed into the mobile device without the need for other abstraction layer to the OSs. They are able to call built-in functionalities such as contacts, dialer, and other types of functionalities. Despite the fact that native applications development requires platform specific skill and expertise, this strategy delivers a higher quality user experience than other MAD methods. Native applications are usually distributed through an application proprietary store (App Store). Figure 2.1 shows the design flow of a native implementation.

2.3 Types of Cross-Platform Tools

The CPTs can be a solution to the platform fragmentation problem because they allow developers to create applications for multiple platforms from almost the same code base or from within the same design tool. This section discusses why CPTs can make a difference in the phase of

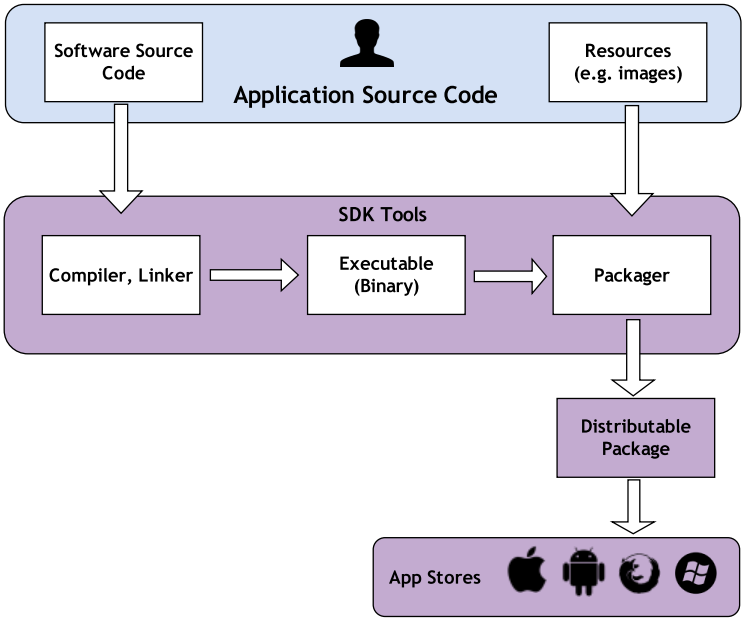


Figure 2.1: Schematic of the design flow for native application development.

application development. It describes the several currently available CPTs along with their advantages and disadvantages (section 2.3).

There are several tools for multi-platform development solving the problem of OS and device fragmentation in the mobile market and each has its own advantages and disadvantages. In the book *Professional Cross-Platform Mobile Development in C #* [S. 12], authors talk about three technologically different approaches, depending on the requirements of cross-platform and application use case: (i) Hybrid applications; (ii) Interpreted applications; and (iii) Cross compiling applications.

Hybrid applications are a combination of the native and web approaches and try to deliver a best-of-both-worlds architecture. Figure 2.2 gives an overview of the architecture of an Hybrid application. They consist of a native wrapper build using HTML, CSS and JavaScript. When executing the Hybrid application, this wrapper code will start a basic web browser (or web view) and will load the HTML, CSS and JavaScript code. For the end user, Hybrid applications look just like a native application, but developers can use HTML and CSS for styling and basic animations and JavaScript for program procedures, while having access to hardware features as with native programming. PhoneGap (see below) is one of most popular CPTs delivering Hybrid applications.

Interpreted applications use run-time environments as a layer between the native OSs to abstract the differences between platforms. The environment can change in size and complexity, and execute code on the device resorting to different technologies as virtualization, interpretation, Just In Time (JIT) compilation or ahead-of-time compilation [Cro12]. Sometimes, the developer

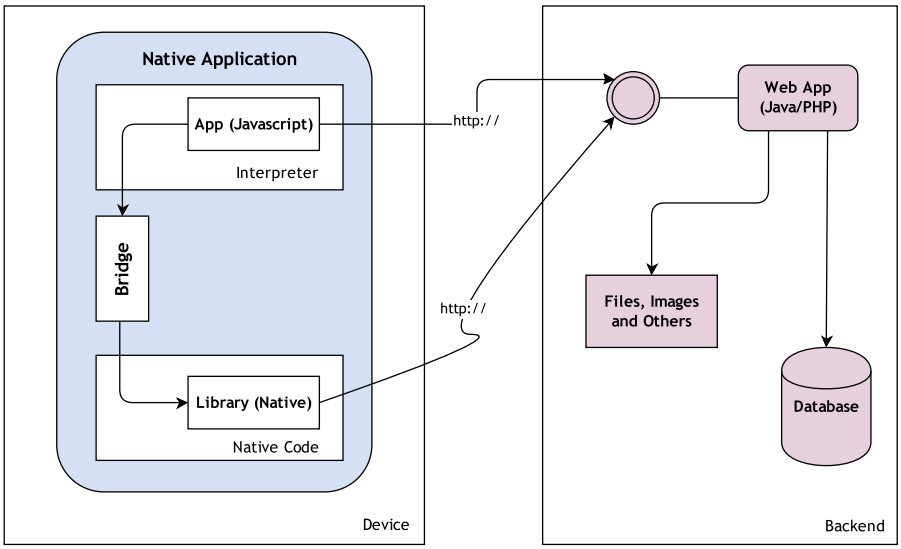


Figure 2.2: Representation of an Hybrid application architecture.

can write code using a well know language like JavaScript for these applications, which will lower some barriers for web developers. The written code will be compiled to some byte-code at design-time and interpreted at run-time, while executing the application. Figure 2.3 shows the architecture of an Interpreted application [Wil12]. A popular tool with this architecture is the Appcelerator Titanium. Cross-compiling tools are, essentially, translators that translate

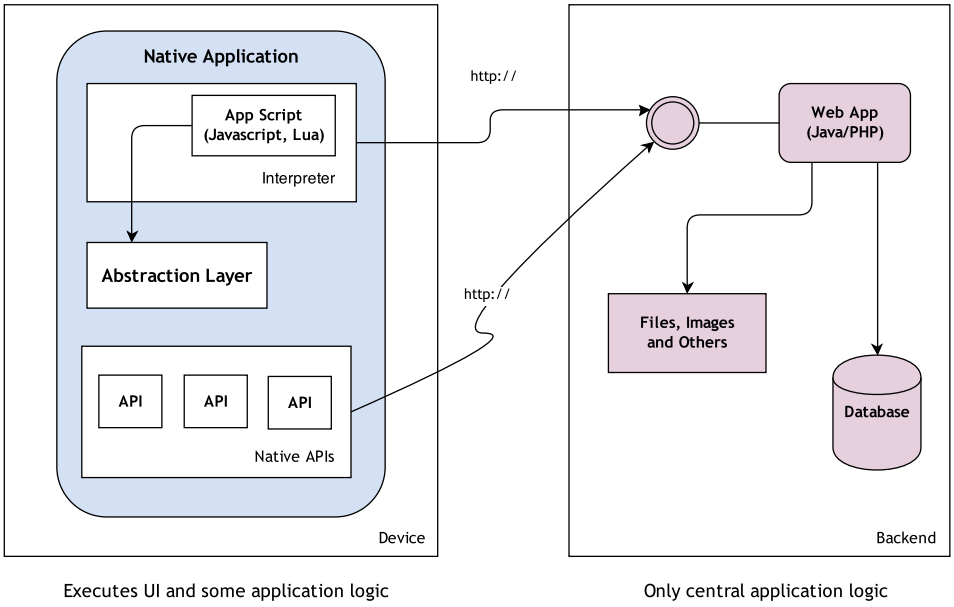


Figure 2.3: Schema of the architecture for an Interpreted application.

code into the native programming language of the mobile platform, intermediary byte code or directly to lower level machine code (assembly language). These tools can often also work in combination with a run-time environment. The Xamarins Mono is a tool that uses this approach.

According to the web page `developer.nokia.com`, the ideal scheme to represent Native development, Hybrid and Web-based (not previously mentioned) applications is the one represented in Figure 2.4. On the other hand, Web-based applications are typically made available in a server over the Internet, and are usually programmed in Hypertext Preprocessor (PHP) and JavaScript, with their interface formatted in HTML5. Many web pages already use this approach, making the website responsive to the device screen size.

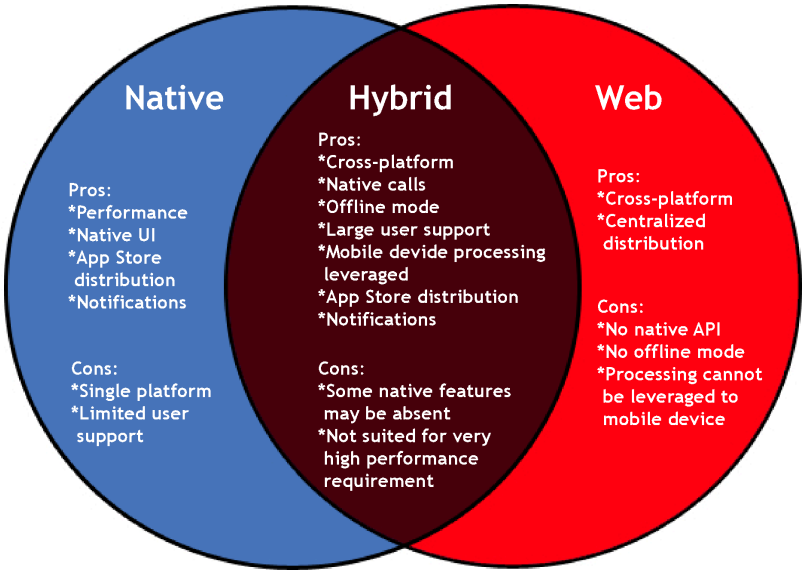


Figure 2.4: Pros and cons of Native, Hybrid and Web-based applications [Nok13].

Advantages and Disadvantages of CPTs

Within the subject of CPTs, some of the most pertinent questions concern the differences between the native and the cross-platform approaches. In general, native development means using the platform-specific tools provided for the mobile OS. In the case of Android, for examples, this normally means using Java and the Android SDK. On the other hand, mobile cross-platform development approaches are more varied in terms of technologies or methodology. There are web applications that run inside of a browser (either standalone or embedded into a container, to more closely mimic a native application). However, there are also approaches that include their own runtime, like the Adobe Air [Ado] (mentioned in this section) and tools that allow programmers to write the code in an abstracted scripting language, and then generate native code at compile time (e.g., the Corona SDK [Cor14]).

According to the Developer Economics website [Dev14] and to the results obtained regarding the mindshare of CPTs (the results were out in January 2014), PhoneGap is the tool kit with greater acceptance by the development industry. As with any development strategy, there are pros and cons to taking a cross-platform approach to design and develop mobile applications. Figure 2.5 illustrates the market acceptance of the toolkit. In the figure, PhoneGap, Appcelerator and Adobe AIR make up the top 3 of the most used cross-platforms in the mobile industry.

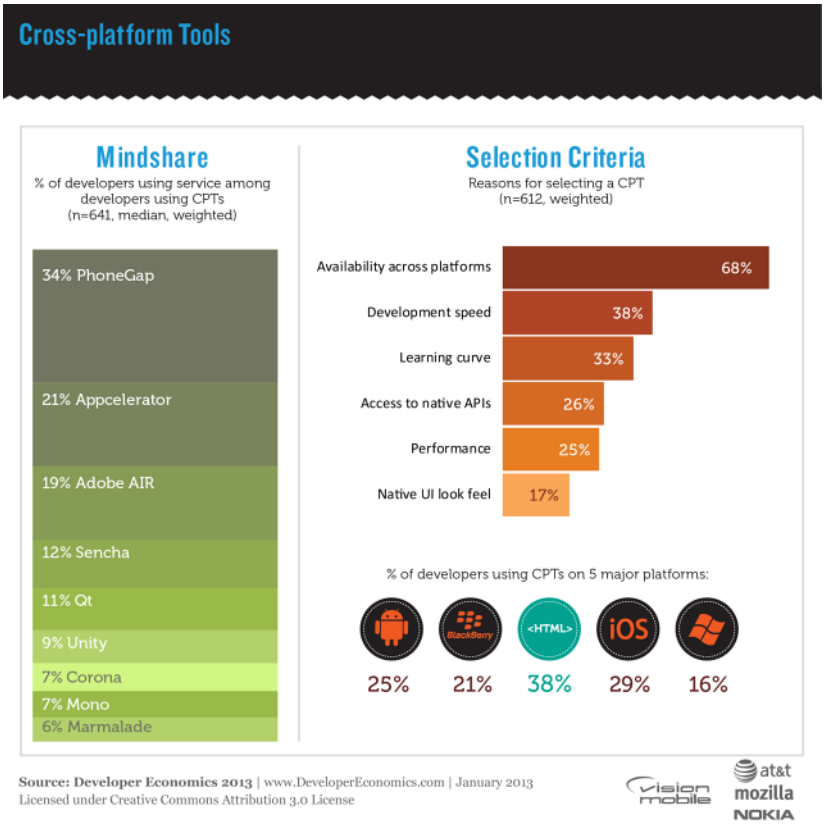


Figure 2.5: Graphical representation of the mindshare of CPTs according to Developer Economics [Dev14].

A recent research from the research2guidance [Res13] revealed that are over 90 CPTs on the market split into 5 categories: (i) Application Factories; (ii) Web Applications ToolKits (or rather, CPMDs); (iii) Cross-Platform Integrated Development Environments (CPIDE); (iv) CPIDE for Enterprise; and (v), Cross-Platform Compilers and Cross-Platform Cloud Services. Figure 2.6 shows the data relative to developers mindshare of this research. The chart in the figure shows a

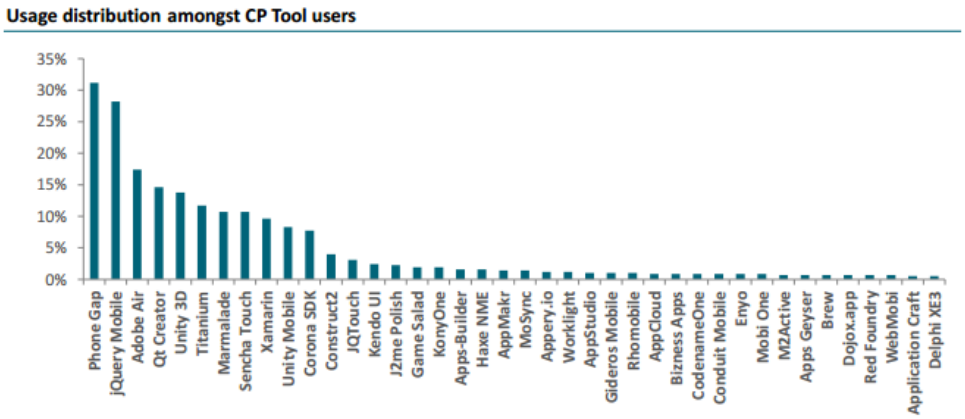


Figure 2.6: Chart summarizing developers mindshare regarding CPTs according with the data presented by research2guidance [Res13].

line up of CPTs different from the one that was previously presented, due to Developer Economics [Dev14]. Even though PhoneGap is still positioned in first, the second most used tool is

now Jquery Mobile. Adobe air maintains its third position in this survey. In the next sections, the pros and cons of the most popular CPTs (according to the Developer Economics research) are presented.

PhoneGap

PhoneGap is an open source framework that provides a decent toolbox for building native mobile applications using only HTML, CSS and JavaScript. Applications built with this framework run in a web view inside a native application container on the target platform. Basically, they consist of a package that allows remote calls to the native API of the device. These remote calls are achieved via JavaScript. Adobe owns the project and offers integration of the framework in its web development tool Dreamweaver [Ado13a]. Apple Xcode IDEs has some neat features that useful to PhoneGap developers as well, namely HTML and JavaScript code highlighting and code completion [Phoa]. PhoneGap is growing fast and new versions are frequently released. The release schedule is necessary to reflect updates in the many platforms it supports and to introduce new functionalities to the enthusiastic community, such as the Adobe PhoneGap Build [Ado13b], a service that allows compilation in the cloud, aiming to simplify the usually complex setup required to compile applications for each platform. The principal advantages and disadvantages of the PhoneGap are mentioned in table A.1, included in the appendix A.

PhoneGap is the most popular CPT in the development of mobile platforms [Dev14]. The JavaScript is the core language in which PhoneGap relies on. The source code is compiled at run time in memory, unlike native platform languages such as Objective C [Appb], Java and C++, which are compiled into executable files. The JavaScript parts of PhoneGap applications source code are distributed in the application bundle and visible to developers after downloading it. It represents kind of a paradise for developers with an open mind. While this can be troubling, it also defines the place of PhoneGap in the software ecosystem. The idea behind software like this will not fade aways easily, which may dictate the success of this product in the future. On one hand, JavaScript is one of the most popular languages nowadays and, on the other, PhoneGap is owned by Adobe, a large company, which also owns Flash technology. This will probably keep PhoneGap in the software landscape for a long time. Moreover, it could make it the right choice for HTML5 application development.

Appcelerator

Titanium (commonly called Appcelerator) [Tit] is a product from Appcelerator that allows developing of applications for deployment on both desktop systems and mobile devices. It allows to create native applications (mobile and desktop) using web technologies, such as JavaScript, HTML and CSS, providing a rich API and low level objects like Transmission Control Protocol (TCP) Sockets. UI objects are customizable through a JavaScript API. The tools present good API documentation but the SDK tools usage (compiler, etc.) are not described with detail. It embeds the

SDK and tools. The generation of native code and the packaging to a native application is made resorting to a Python script (for Android). The Appcelerator advantages and disadvantages are provided in table A.2, included in the appendix A.

Adobe AIR

In accordance with the website Developers Economis [Bro], Adobe AIR [Ado] is a cross-operating-system runtime that lets developers combine HTML, JavaScript, Adobe Flash and Flex technologies, and ActionScript to deploy Rich Internet Applications (RIAs) on a range of devices that include desktop computers, netbooks, tablets, smartphones, and Televisions (TVs). The pros ans cons of this tool can be summarized in table A.3, included in appendix A.

Sencha

Sencha [Sen14] Touch is a comprehensive JavaScript framework. It provides facilities for programming user interfaces, business logic and data access and has a large number of utility classes. This is an HTML5 mobile application framework for building web applications that look and feel like native applications. Applications built with Sencha can be used with PhoneGap, either which will package the application in a native container and enable access to select device-level APIs, topically unavailable to traditional web applications. Ir is the major competitor of Jquery Mobile. In order to understand its potential, some advantages and disadvantages of this CPT are presented in table A.4 in appendix A.

Qt

Qt ("Cute") [QT14] is a CPT that targets a number of embedded, desktop and mobile platforms. With Qt, developers write applications using Qt Modeling Language (QML), a CSS and JavaScript like language [Qt]. Applications are backed with an extensive set of C++ libraries, and utilize graphical UI components written in C++. To summarize its power and penetration in the mobile development industry, table A.5, in appendix A, presents its advantages and disadvantages.

Corona

Corona SDK [Cor14] is a cross-platform technology that utilizes Lua scripting to enable abstracting the application development from a specific mobile OS. The application can be tested on a simulator and then compiled into native code for Android or iOS. The table summarizing its advantages and disadvantages is table A.6 of appendix A.

Mono

Mono (MonoTouch) [Mon14] is a framework that allows developers to create iOS applications using a collection of existing C# and .NET source code, libraries and procedures. Though it is not a single tool, it was considered that it could fit in this section. Its advantages and disadvantages are presented in the table A.7, included in appendix A.

JQuery Mobile

The JQuery [JQu14b] Mobile is a cross-platform mobile framework designed to simplify and enhance the development of mobile web applications by integrating HTML, CSS, JQuery and JQuery UI into one framework that is not only robust, but maintainable and organized. This tool and Sencha Touch can be combined for mobile development, handing their functionality to tools like PhoneGap. Table A.8, in appendix A, presents the JQuery Mobile advantages and disadvantages.

2.4 Benefits of CPMDs

With the objective of easing mobile application development, web based platforms with embedded CPTs in their structure have been created. In a way, they create an even higher layer of abstraction in mobile development. Tools like PhoneGap are included in this type of frameworks. There is already a wide range of this type of platforms currently available on the market. The design and structuring of these frameworks are simple, using the drag-and-drop of objects as the key concept to facilitate construction and editing of new applications.

This masters programme is precisely focused on the creation of a CPT, like PhoneGap, and incorporate it into a web platform for mobile development. From the combination of both of them germinates a complete system for independent development of mobile applications, i.e., a CPMD. This new platform will not use any type of the available CPTs on the mobile industry, comprising thus a potential free alternative for this class of frameworks.

CPMDs promise lower cost and reduced time of development, since they leverage the use of the same code base to maintain and write applications, while targeting multiple devices and platforms. Figure 2.7 illustrates the trade-offs between the combinations of time with cost and UI with performance for the several types of applications discussed herein. In an article

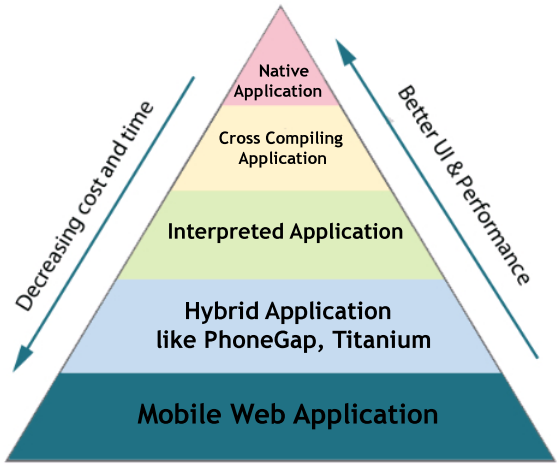


Figure 2.7: Trade-off between cost, time, performance and UI for the several types of mobile applications, in terms of development.

about mobile development [Dam11], *D. Gavalas* and *D. Economou* claim that cross-platform development might inhibit future development since it then might be more focused on making it easy for developers, instead of striving for innovation. Some noteworthy benefits of CPMDs are as follows:

- Cross-platform development is mainly based on HTML5 and related technologies, and web developers might be able to leverage their experience;
- Learning web technologies might be less challenging that learning a new platform of development (API);
- Devices pack all sorts of technology, specifications and features, and cross-platform development might decrease the effort of making an application work in many different combinations of those axis;
- Resorting to CPMD means using only an SDK and an OS, which means less initial investment;
- These platforms are getting increasingly better at emulating natively written applications;
- A non-native application might exploit the native controls well;
- The usage of CPMD might help reduce costs in human resources and the development cycle;
- They might help developers to reach mass devices through their hardware and platform abstraction features.

2.5 General Architecture of CPMDs

Developing an application that can run on multiple platforms such as iOS, Android, RIM, Symbian, Windows Phone and Bada or others, might be a difficult task, considering the different technology and specific artifacts of each one of them. The development objective can be either native or cross-platform application. The MADs approach can follow may use HTML5 and CSS for the definition of the UI, while JavaScript, C++ and Java are the preferred the languages for the programming logic. During MAD phase, it is common to use IDEs , debuggers, testing modules (device emulators or simulators), and source control tools [Vis12, Ada11]. It is at this point that CPMDs established the bridge between the mobile applications and CPTs. These CPMDs are web-based frameworks of development that allow implemented mobile applications to run on a wide variety of different mobile OSs. The combined web technologies and native API accessibility features of a mobile device. These applications are written mostly in HTML, CSS and JavaScript. Figure 2.8 shows the general architecture of a CPMD, where the connection and role of the technologies emphasizes. This type of platforms [Bro, The12, Gooa] provide a first

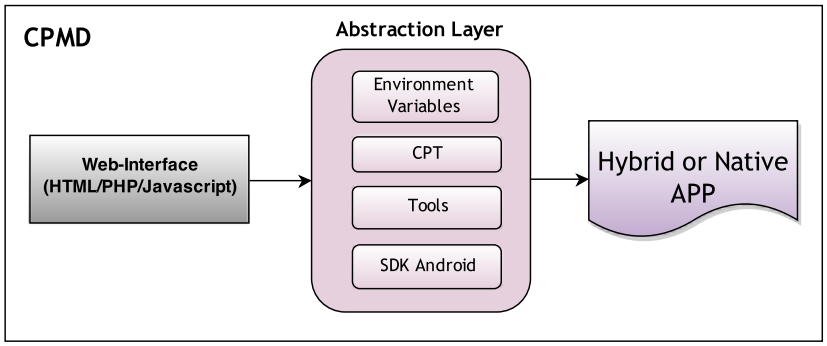


Figure 2.8: Representation of the general architecture of a CPMD.

layer consisting of the communication interface with the user. It mostly uses HTML5 combined with JavaScript and PHP, to provide the context for creation and development of applications. The abstraction layer aggregates a set of features, environment variables, one or more CPTs, and some essentials tools for the functional of the platform. It makes the direct linkage with the SDK, to allow the generation of mobile applications. Two of the most important components, in this context, are the CPTs and the SDKs. These components are the soul of this type of mobile frameworks. CPTs, as mentioned throughout the dissertation, allow cross-platform development, usually making use of tools like Phonegap. In this masters programme, an integrated CPT was built for scratch, and embedded in the framework (this subject is further explained in chapter 4). The SDK enables the creation and translation to native code, finally producing fully native or hybrid applications, according to the context and purpose of the development framework.

2.6 Comparative Study of CPMDs in the Market

The development process involves having a profound insight into the software target users, the target platform, the programming languages and the tools. There are also methodologies or frameworks that could potentially be considered to manage and plan the process. Expertnear framework, the name given to the proposed CPMD, is one of them. It may constitute a great aid in the phase of creating and planning new mobile applications. It offers a set of features and UI designs to wide range of contexts.

In order to provide a perspective regarding the placement of the Expertnear framework and relation to other solutions in the market, a series of tables, resulting from this study and comparing them, is included below. The CPMDs used in this comparison are the ones mentioned in the Mashable web page [10 13]. Some of them are very similar. The several tables summarize comparisons for (i) supported OSs; (ii) their features; (iii) the used CPT; (iv) the industry focus; (v) the available support services; and (vi), the license type of each framework.

Overall, all the frameworks described in table 2.1 can be used to export a project for the most

common types of mobile OS, namely, iOS, Android and Windows Phone. Nonetheless, only the framework Expertnear possesses a module to export applications for the FirefoxOS.

Table 2.1: Types of mobile OSs supported by the frameworks.

	iOS	Android	Windows Phone	HTML5	FirefoxOS
appery.io	×	×	×	×	
mobileroadie.com	×	×		×	
theappbuilder.com	×	×	×	×	
goodbarber.com	×	×		×	
appypie.com	×	×		×	
appmachine.com	×	×	×	×	
gamesalad.com	×	×	×	×	
biznessapps.com	×	×		×	
appmakr.com	×	×	×	×	
shoutEm.com	×	×		×	
como.com	×	×		×	
Expertnear		×		×	×

Android and iOS systems are supported by the majority of the frameworks. Microsoft Windows Phone OS is less supported. Table 2.4, which presents a comparative study of the features and modules of each system, was build after studying each platform individual. Due to their size, some of the tables were rotated and included in the end of this chapter. As can be observed in table 2.2, the most common CPT is PhoneGap. This fact corroborates the results of the study illustrated in figure 2.6 and also figure 2.5.

Table 2.2: Identification of the most popular CPT of each CPMD.

	CPT(s)
appery.io	PhoneGap and Jquery
mobileroadie.com	Integrated
theappbuilder.com	?
goodbarber.com	?
appypie.com	PhoneGap
appmachine.com	PhoneGap and Jquery
gamesalad.com	?
biznessapps.com	Appcelerator and PhoneGap
appmakr.com	PhoneGap
shoutEm.com	PhoneGap
como.com	?
Expertnear	Integrated

? represents the absence of information.

The majority of CPMDs are not focused on delivering applications for particular business and industrial contexts. However, the scope of some of the frameworks is limited to sectors such as *Retails*, *News* and *Sports*. The industry focus of the several CPMDs summarized in table 2.5.

CPMDs vendors offer a panoply of support options, ranging from online community support to *on site* services. The most common support channels are online community support and time-delayed online support. In all cases, support services are available in English, but some vendors

of CPMDs offer their support services, in additional languages. For a few frameworks, support is offered in a number of widely-spoken languages. Most of the frameworks concentrate more on online support channels, whereas more complex solutions allow *on-site* training and project support. Table 2.6 is devoted to presenting this part of this study. The CPMDs studied throughout this dissertation are licensed frameworks, just with a trial period for new users. Notwithstanding, these kind of users have the possibility to try out and enjoy the resources of the system. Table 2.3 displays data regarding licence types for the several CPMDs.

Table 2.3: Licence type and usage period of each CPMD.

	Free	Trial Period	Commercial
appery.io		×	×
mobileroadie.com		×	×
theappbuilder.com			×
goodbarber.com		×	×
appypie.com		×	×
appmachine.com		×	×
gamesalad.com		×	×
biznessapps.com		×	×
appmakr.com		×	×
shoutEm.com		×	×
como.com		×	×
Expertnear	×		

2.7 Conclusion

This chapter introduced various concepts related with CPTs and CPMDs. Some of the existing CPTs in the mobile industry were briefly described at the beginning. Their advantages and disadvantages are included in a tabular form in appendix A. As emphasizes this chapter, these tools are the core of any CPMD, since they support the link between the application and the native components of the mobile devices. It was possible to conclude that the PhoneGap is the most used tool in the cross MAD, mostly because it is a free tool. The last part of the chapter presents an overview between web platforms for developing mobile applications, which are commonly named CPMDs. The demand for these frameworks is high nowadays. For banks, start-ups, convenience stores, government organizations, etc., mobile applications and frameworks to build them representing a solid means to get the to the masses. A comparison of the performance and behaviour between native and non-native applications is not performed in this chapter, but it will be provided in chapter 5.

Finally, it can be said that mobile development is a very hot topic nowadays, with a highly dynamic and vibrant community of users and solution providers. Although experience constitutes a major factor when choosing the right approach for mobile development, things change so fast in this field that what was a best practice a few years ago can become irrelevant as new technology breaks through.

This chapter provided the context for CPMDs. The following chapters describe the engineering and development and the proposed framework. The security mechanisms to assure reliability and transparency is also subject of discussion, as well as some interesting implementation details.

Table 2.4: Comparison of the characteristics and features of CPMDs.

	Multi- Languages	Drag-and-Drop	Social Networks	QR-Code	Full-Support	API Access	Security Modules	Project Shared
appery.io	×	×		×		×	×	×
mobileroadie.com	×		×	×	×			×
theappbuilder.com	×		×				×	
goodbarber.com	×		×					
appypie.com	×	×	×	×	×		×	
appmachine.com	×		×	×				×
gamesalad.com	×	×			×	×	?	×
biznessapps.com	×		×		×			
appmakr.com	×	×	×		×		×	
shoutEm.com	×		×			×		×
como.com	×		×	×		?	?	
Expertnear		×	×	×	×		×	×

Table 2.5: Industry and sector focus of CPMDs.

	General	Games	Retail	Hotels	News	Sports	Music	Financial	Education	Enterprise	Image Processing
appery.io	×		×		×					×	
mobileroadie.com	×		×		×	×	×			×	
theappbuilder.com	×		×	×	×	×	×	×		×	
goodbarber.com	×				×	×	×			×	
appypie.com	×		×	×	×	×	×	×	×	×	
appmachine.com	×		×		×	×	×		×	×	
gamesalad.com	×	×									
biznessapps.com	×		×	×	×	×	×	×	×	×	
appmakr.com	×		×	×	×	×	×	×	×	×	
shoutEm.com	×		×	×	×	×		×	×	×	
como.com	×		×	×	×	×	×	×	×	×	
Expertnear	×		×	×	×		×	×	×	×	×

? represents the absence of information.

Table 2.6: Available support services and languages of CPMDs.

	On-site support	On-site tutorials	Personal phone	Real time support	Time-delayed support	Others	Multi-Language
appery.io	×	×			×	blog, videos	×
mobileroadie.com	×	×					×
theappbuilder.com	×	×	×	×		blog, videos	×
goodbarber.com	×		×	×			×
appypie.com	×	×			×		×
appmachine.com	×	×			×		×
gamesalad.com	×	×	×		×		×
biznessapps.com	×	×				blog, videos	×
appmakr.com	×	×	×	×		blog, videos	×
shoutEm.com	×	×			×	blog	×
como.com	×	×	×	×		blog, videos	×
Expertnear	×						

Chapter 3

Software Engineering

The previous chapter ended up with an overview on automated tools for mobile development. This chapter focuses on the software engineering of the CPMD developed along this masters programme, resorting mostly to Unified Modeling Language (UML) concepts and diagrams that may help to structure the development phase and describe what was done. Whenever pertinent and possible, security aspects taken into account during the planning of the framework will also be discussed herein.

3.1 Introduction

Planning and modeling a software is one of the most important phases of its life cycle. It involves identifying its functional and non-functional requirements (section 3.3), its use cases (section 3.4) and main Activities (section 3.6). If the software is to be developed using an object oriented approach, it is useful to determine the Classes (section 3.5) and their relations in this phase also. Moreover, if a database is to support the system, it is at this stage that it should be modeled and documented, e.g., using entity-relationship diagrams (section 3.8). The following section briefly describes the main objectives of the engineered software, so as to provide the context to the remaining part of the chapter.

3.2 Objectives

As mentioned in the *Problem Statement and Objectives* of this dissertation (section 1.2), one of the objectives of this masters was to develop a web platform for creating web and mobile applications. Such platform should provide users without knowledge in programming with the means to develop applications in an intuitive manner. The idea is to hand out a prepared set of features that they may jiggle around in a *what you see is what you get* Graphical User Interface (GUI) to design new mobile applications. The abstraction level offered by the platform should enable the user to only care about the aesthetic definition of the application and full support at each step of development should be provided. The aforementioned steps should be delimited in well organized development modules and accompanied by personalized help to the user. It is important to make an effort in simplifying all the processes involved, given that such effort will impact the intuitiveness and ease of use of the platform, ans should be present since the planning and engineering phase. It should be mentioned that the tool should furthermore enable users to create both native and non-native applications, i.e., HTML5 and FirefoxOS based applications

and Java-based applications (for the Android platform). Given this general statement concerning the objectives, it is now possible to identify the general requirements of the framework.

3.3 Requirement Analysis

The requirements analysis provides the specification of what should be implemented in the software system. They are typically divided into functional and non-functional requirements, depending on whether they describe the behaviour that the system offers to the user or the conditions in which it should operate, respectively. The functional requirements identified for the framework developed within the scope of this masters are enumerated below:

- The framework should contain one page dedicated to the promotion and presentation of the framework, in order to attract the interest of new users;
- Support for two types of users should be provided, namely for an administrator (who manages the whole framework) and for a user (person who uses the framework to build mobile applications);
- Unregistered users should be able to see the applications from other users in a dedicated part of the framework, but they should not be able to edit those applications;
- After registration, all users of the framework should be presented with an optional tutorial to create a new application;
- The recovery of the personal password and the method of authentication must be secure by design;
- The user (owner of a project) can assign other users to his or hers project, to motivate parallel development;
- The user should be able to create a team and, in a reserved area, assign tasks to other members, define deadlines and send personal or global messages to other users;
- After creating an application in the framework, the user must be able to assign a password to access that application;
- When creating the cross-platform application, the user should be asked to define the name, purpose (business or personal), type (such as music, information, business, etc.) and the privacy policy;
- The user must have the opportunity to customize the cross-platform application, the background, layout, text boxes, buttons, etc.;

- The opportunity to import external content into the cross-platform application, as for example, youtube videos, pictures, etc., should be presented to the user during the creation of an application;

- The user must be able to create a homepage and browse through other pages, which should also allow a custom setting;

- The platform should provide default features, e.g., style menus, information pages, information pages with images, news pages, contact pages, etc.;

- A personalized and non intrusive help or tutorial system must guide the user through all of the steps;

- The user must be able to export the project to HTML5, Android and FirefoxOS;

- Exporting, deploying and installing the applications generated by the framework should be made simple via the usage of already established Quick Response (QR) codes technology.

Notice that the translation of code for iOS was not tackled throughout this work. It was decided to focus on platforms and technologies that were more readily available, namely open-source or free solutions.

In terms of non-functional requirements, and since this platform should necessarily operate on the client-server architecture, it is important to distinguish the constraints for each entity in the system (server and client). The remaining part of this section is thus structured into two parts.

Non-functional requirements of the server

The server must permanently run an Apache server with the Transport Layer Security (SSL) module, alongside with a PHP processor and a Data Base Management System (DBMS). It must also have an updated Android SDK installation. At the hardware level, and assuming that no flash crowds to the framework are expected to happen, this machine must be able to support some connections simultaneously and must be able to respond efficiently to all of them. With these requirements, the machine must have at least the following specifications:

- It must possess at least 2GB of Random-access Memory (RAM);

- The Hard Disk must have at least 160GB (500GB is advised for guaranteeing operation in the long-term);

- It must at least have a state-of-the-art 2.8GHz quad core processor;

- It must be connected to the local computer network via a 100Mbps card and to the Internet;
- There is no specific constraints regarding the OS, as long as it supports the aforementioned software. The OS may be any server oriented Linux distribution or Microsoft Windows.

Non-functional requirements of the client

As the interaction between the user and the system will be performed through a web-browser, the list of specifications is simpler. The framework should be accessible to any modern computer or device with the von-Neumann architecture, capable of running an up-to-date web browser with JavaScript support. Mobile phones, smartphones, tablets, desktops and laptops constitute examples of devices from where the framework can be accessed from.

3.4 Use Cases

An Use Case is an autonomous activity typically performed by an actor of the system. The process of identifying use cases comprises one of the most basilar tools of software engineering. The set of Use Cases identified for the proposed framework is included in Table 3.1, which summarizes the more detailed use case diagrams included in Figures 3.1, 3.2 and 3.3.

Figure 3.1 shows the most general use case diagram of the system, from which it is possible to derive the main functionalities it should provide, as well as the stakeholders and their actions in different usage scenarios. Some of the most important details and statements depicted in the diagram are as follows:

- The diagram shows that 3 privilege levels are being considered (the administrator, the normal user and the external user privilege levels). In the requirements analysis, it was hinted that the developed system would distinguish two types of user: the administrator and the regular user. Nonetheless, in the represented use cases, it was considered also the non-registered user, which may access only very basic functionalities of the platform;
- The administrator and the registered user need to be authenticated in the system prior to be able to use its functionalities;
- The administrator is able to manage all users;
- The user manages projects and can coordinate a development team for each project;
- An external user can only see projects created by registered users in the platform, but not edit them.

The *Create Project* use case is then presented with more detail in figure 3.2, while the *View User Projects* use case is further elaborated in figure 3.3.

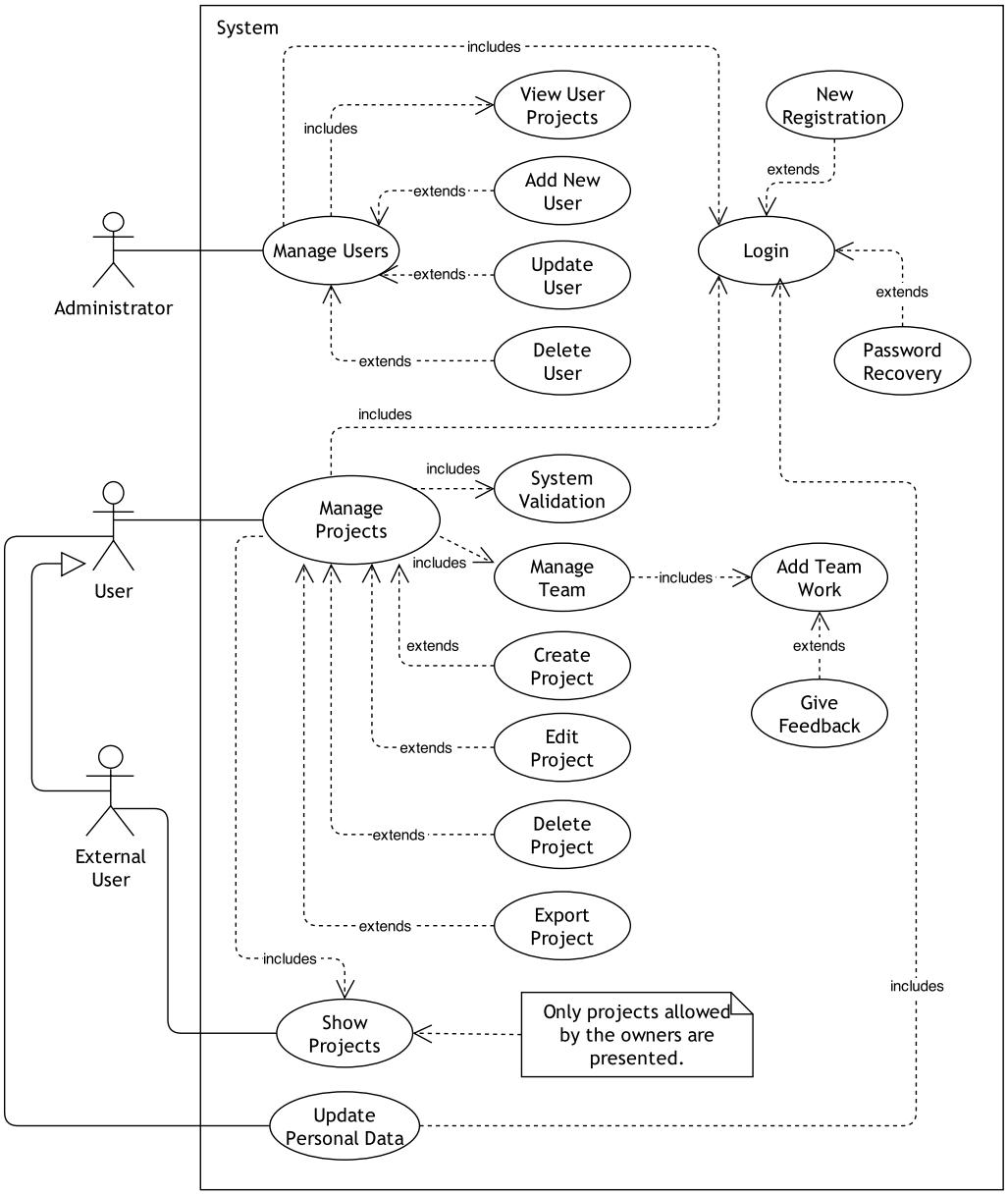


Figure 3.1: General use case diagram.

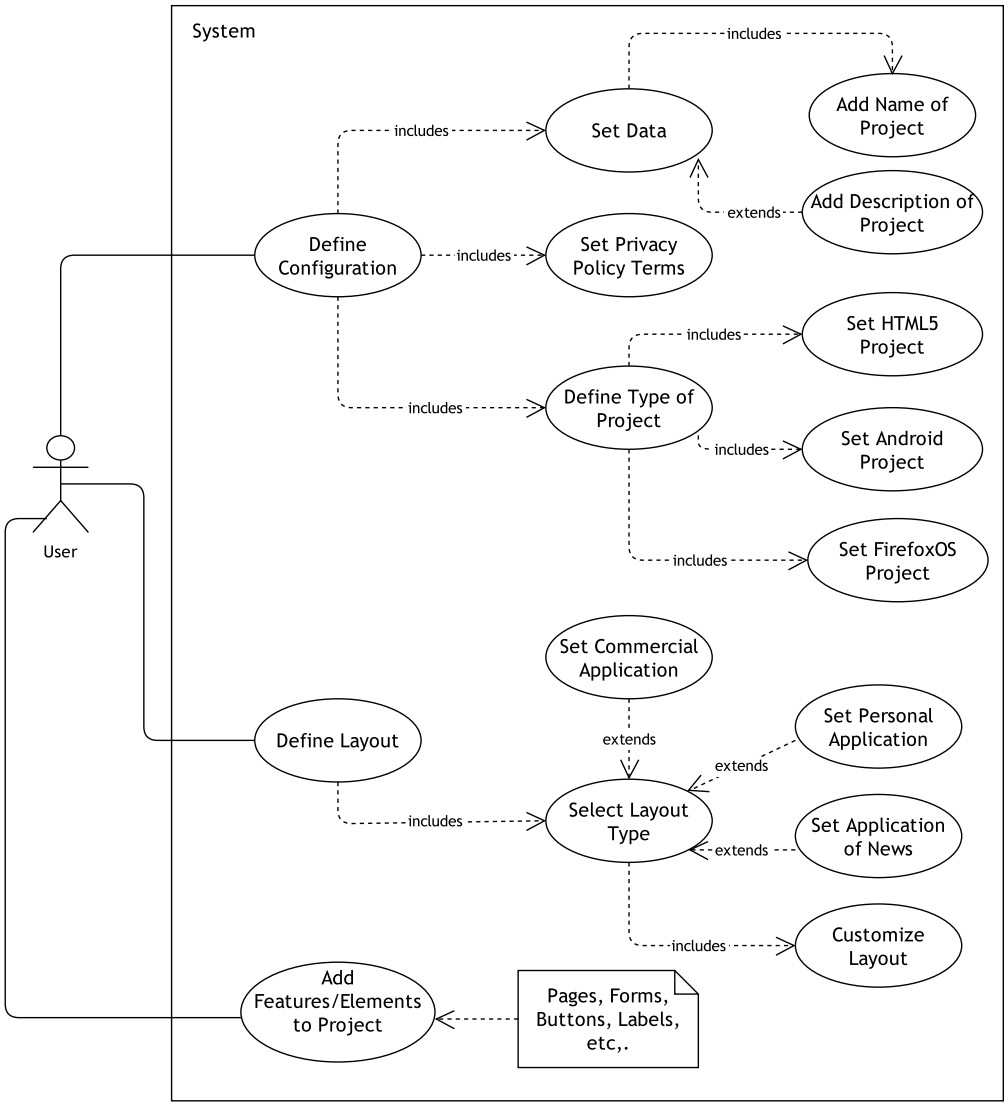


Figure 3.2: Detailed use case diagram for the more general *Create Project* use case.

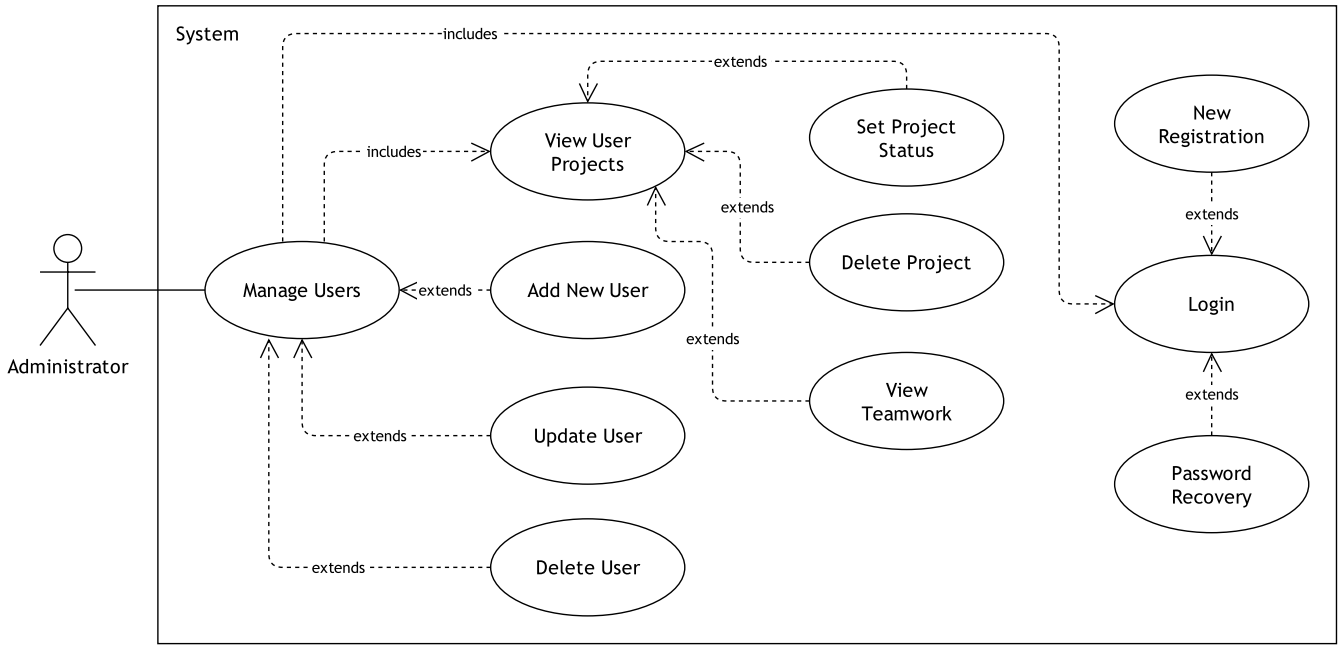


Figure 3.3: Detailed use case diagram for the more general *View User Projects* use case.

Table 3.1: Identification of the use cases of the framework.

Actors	Use Cases	Abstract Cases
Administrator	Login in the system Manage Users View User Projects	View Teamwork Add New User Update User Delete User Set Project Status Delete Project
User	Login in the system Manage Projects System Validation Manage Team Add Teamwork Show Projects Update Personal Data Define Configuration Define Layout Select elements to layout Set Data Set Privacy Policy Terms Select Layout Type Customize Layout	New Registration Recovery Password Create Project Edit Project Delete Project Export Project Give Feedback Add Name Of Project Add Description Of Project Set HTML5 Project Set Android Project Set iOS Project Set Commercial Application Set Personal Application Set Application Of News
External User	Show Projects	

In the software engineering process, the discussion of some scenarios follows the identification of use cases. The scenarios are the paths that can be traversed to perform an activity, task or requirement. Main scenarios assume that all conditions are met to ensure that all is well, and side scenarios define alternate paths regarding to wrong or improbable situations in the main scenario. The discussion of several scenarios for each one of the identified actors and some use cases was included in appendix B. This discussion was specially useful during the development of the framework, aiding mostly in the definition and re-thinking of the workflow.

3.5 Class Diagram

This section discusses the class model devised for the CPMD developed in the scope of this masters, taking the aforementioned use cases as basis. The model is depicted resorting to the typical class diagram in figure 3.5, which was rotated for a better fit on the page. The classes in the diagram capture some of the most important features of the system in terms of attributes and operations and it is possible to conclude from its analysis that special emphasis was placed on security (3 classes), user and team management (2 classes) and cross-mobile development (5 classes). In terms of purposes and objectives, the several classes can be summarily described as follows:

- The `Register` class possesses all variables and methods needed for the registration of new users. Data concerning new users is initially stored in a temporary table until all the registration steps are completed. The methods included are `AddUserTemp`, `SendEmail`,

AddUser, RemoveFromTemp, RecoveryPwd, Injection and Verify.

- The User class is responsible for providing all the methods related with the management of users. It provides the methods CreateUser, ModifyUser, DeleteUser, Injection, Verify and GetUser.

- The Login class is responsible for providing the system authentication methods. The software included is implemented in the methods Authentication, VerifyHoneyWords, Injection and Verify.

- The Team class allows the creation of teams. It has all methods necessary for the creation and interaction between a work group. It includes methods such as CreateTeam, ModifyTeam, DeleteTeam, AddUser, DeleteUser, SendEmail, Injection, Verify and GetInfo.

- The PBKDF2 class contains all the methods responsible for building an iterative procedure to create the hash key derived from the password of the user for system authentication. The respective methods of the class are CreateHash, ValidatePassword, SlowEquals, RandomNonce and PBKDF2.

- The Project class allows the user to create, edit and manage his or hers projects. The main methods it includes are CreateProject, ModifyProject, DeleteProject, Injection, Verify and getData.

- The Development class offers the methods responsible for in building a new cross-platform application. It implements the methods DragandDrop, RemoveFile, EditFile, Refreswebview,Injection and Verify.

- The SecureMethods class implements some security related mechanisms, most of them with the objective of protecting against pre-computed injection code, session highjacking and also the Honey Words mechanism [Ari13]. The methods it includes are SQLInjection, GenerateSalt, StrongPwd, GenerateHoney, GenerateJ, LegitimateSession, Injection and Verify.

- The Layout class provides the system with the necessary methods for the selection of the layout during the creation of a new project. The software included is implemented in the methods AddLayoutProj, GetLayouts, UpdateLauout, Injection, Verify and the GetData.

- The Features class contains the methods implementing the features provided by the framework. The behavior of the class is defined by the methods AddProFeature, UpdatePro

Feature, DeleteProFeature, GetData, Injection and Verify.

- The Export class possesses the set of methods associated with project exportation, including Android, HTML5 and FirefoxOS. Each one of the target platform has its set of methods, which include the TestType, IdentifyType, CreateProjectFirefox, CreateProjectHTML5, CreateProjectAndroid, RemoveProject, CreateNativeAndroid, GenerateQrCode, SendEmail, Injection and Verify methods.

The class mentioned in last (Export) contains a method called CreateNativeAndroid, which is responsible for creating native mobile applications for the Android OS. This particular subject is further discussed in chapter 4. Nonetheless, notice that the engineering of the platform already defines that the CPMD to be developed should contemplate both native and non-native applications.

3.6 Activity Diagrams

This section includes and discusses some of the activity diagrams that allow the understanding of the workflow of some interesting features of the system. They were elaborated starting from the use cases identified earlier for each one of the actors. Since particular attention to the security aspects of the platform was to be paid, two of the four activities discussed herein are related with that specific topic, since they comprise good tools to assure security by design. A larger set of diagrams were produced during the software engineering, but it was decided to include only four, since they are representative of this part of the work. The use cases in which they are based on are: (i) *Login in the system*; (ii) *Manage Project*; and (iii), *Select Elements to the Layout*.

Table 3.2 describes the interaction between the user and the system during the registration activity. The activity concerning the user authentication in the system is then presented in table 3.3. The process involving the creation of a team and project management is scrutinized via the analysis of the activity described in table 3.4, while the process of adding a new feature is discussed and illustrated by the activity diagram included in table 3.5. All tables include a graphical and textual description of the aforementioned activities.

3.7 Component Diagram

The components diagram for the CPMD platform that was developed in this masters is depicted in figure 3.4. Since a CPMD is typically a web-based system, it uses the client-server architecture with the browser taking the role of the client and most of the software to be developed assuming the role of the server. Notice that, in this particular case, there are some functionalities that will be provided via JavaScript, to leverage the power of the computers using the platform, thus placing some software on the browser also. The diagram also shows that it was planned to place

the database in a separated server, with all communications supported by the TCP/Internet Protocol (IP) protocol suite. The proposal to place the database in a different server is mostly due to security reasons. Nonetheless, for the prototype described in the following chapter, the database management system and the server side of the CPMD platform were running in the same machine. While an effort to identify the exact technologies or systems used for the several components was performed and included in the diagram, some of that information was merely indicative.

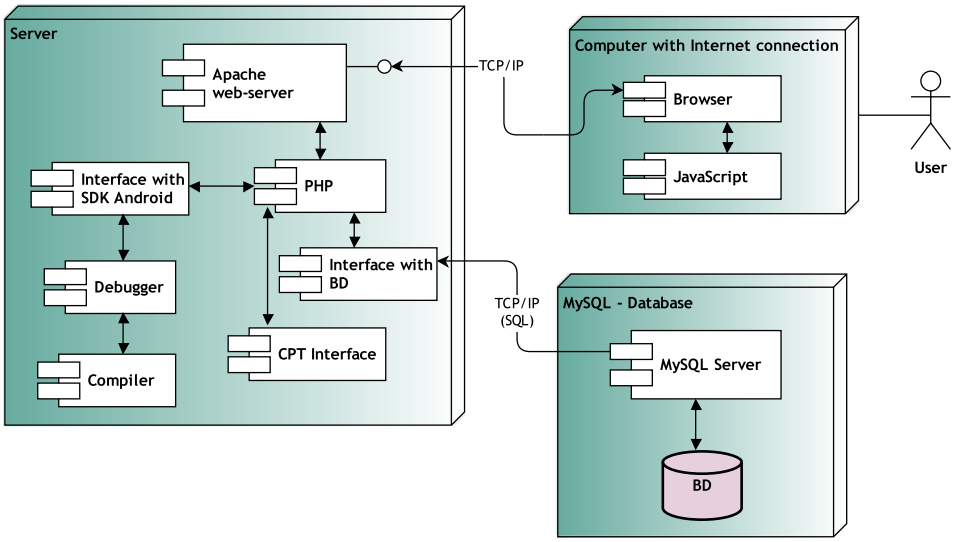


Figure 3.4: The components diagram of the engineered system.

3.8 Entity-Relationship Diagram

Figure 3.6 presents the Entity-relationship diagram representing the database for supporting the engineered system. Databases typically play a key role in web based systems and CPMDs are no exception to that rule. The diagram identifies the entities that model the system from the data point of view, showing also the relations between those entities and the associated cardinalities. A total of 13 entities have associations while 2 represent isolated features (`Promotional Message`) or temporary data structures (`UserTemp`). The `User` is considered the main entity of the database, to which most of the entities are connected.

The importance given to security aspects is emphasized by the database model too. For example, it can be seen that each user is associated with a set of honey words [Ari13] for password and database leakage protection. The `AuthJ` entity is also related with this mechanism, and it represents encrypted information regarding the original user password.

`Project` is another central entity of the system, with a total of 5 relationships with other entities. Notice that this entity will end up being the one storing part of the information regarding applications created by the CPMD.

3.9 Conclusion

This chapter presents the software engineering and the most important requirements for a platform like a CPMD. It identifies the actors and discusses the use cases that determine the functionalities of the system to be developed.

Modeling was mostly performed resorting to UML class, activity and components diagrams. A model for the supporting database was also presented via an entity-relashionship diagram. These elements were used to show that security was taken into consideration since the very beginning, and also to enumerate some of the features to be delivered by the CPMD. The design of the system assumes that support for building native Android applications is to be provided along with the ability to generate non-native mobile applications.

Starting from the guidelines provided in this chapter, the following one can now elaborate on the implementation details of the system developed along this masters. The included activity diagrams are particularly interesting for the explanation concerning the implementation of the security mechanism known as Honey Words [Ari13].

Table 3.2: Diagram and description for the *User Registration* activity in the system.

Use Case name	Login in the System (Registration)
Description	1. The user accesses the registration page of the framework. 2. Then, introduces the data in the form, and its validity is checked in the browser (e.g., using JavaScript). 2.1. If the data is not valid, an error message is triggered, and the user must re-enter the data. 2.2. If successful, the system (server-side) performs data validation too. 3. The system uses protection functions against pre-computed malicious Cross-site Scripting (XSS) and SQL Injection (SQLi) codes. 3.1. If the data is invalid, the workflow will return to the beginning. 3.2. Otherwise, the system creates Honey Words (further explained in chapter 4) derived from the user password. 4. The data is added to the supporting database. 5. The system displays a success message. 6. The user accesses the homepage. Upon registration confirmation, the user can access login area.
Activity Diagram	

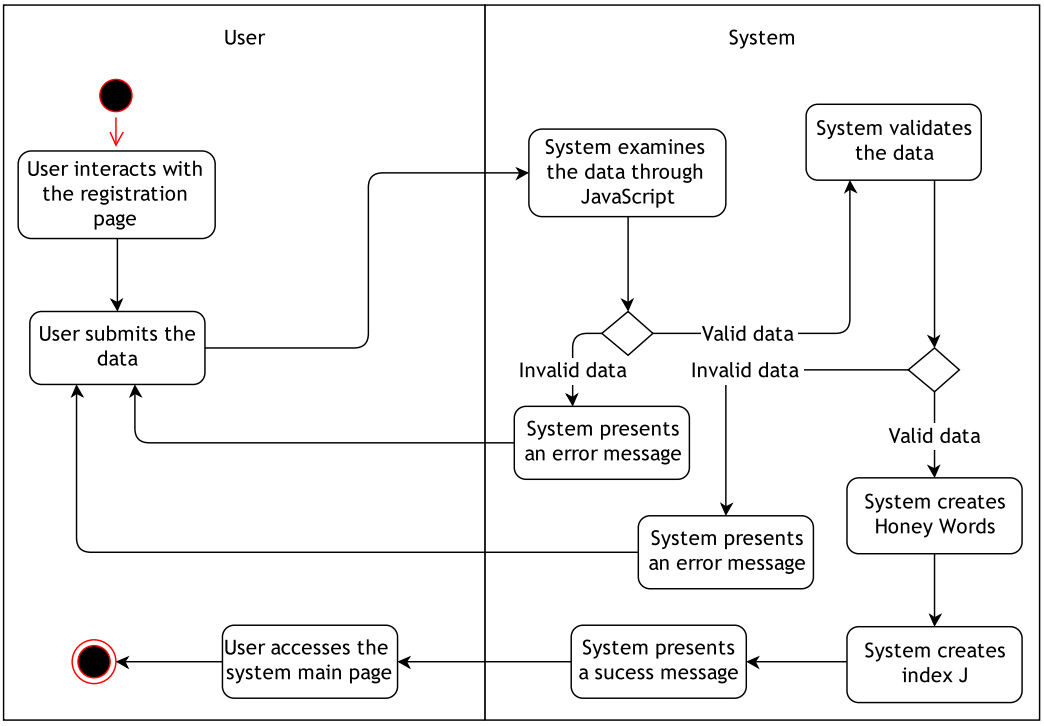


Table 3.3: Diagram and description for the *User Authentication* activity in the system.

Use Case name	Login in the System (Authentication)
Description	1. The user accesses the login page. 2. Enters access credentials (username and password) and the system checks the data using in the browser (i.e., using JavaScript). 2.1. If a problem is detected (e.g., the username does not have a valid format), an error message is triggered, and the form to re-enter the data is displayed. 2.2. If successful, the system (server-side) performs data validation. 3. The system uses protection functions against pre-computed malicious XSS and SQLi codes. 3.1. If the data is invalid, the workflow will return to the beginning. 4. The system will then try to detect if the password is a honey word or not. 4.1. If it is a honey work, an alarm is triggered for the administrator. 4.2. Otherwise, authentication is performed successfully in the system. 5. User is redirected to the home page of the system, where the features for authenticated users are now available.
Activity Diagram	

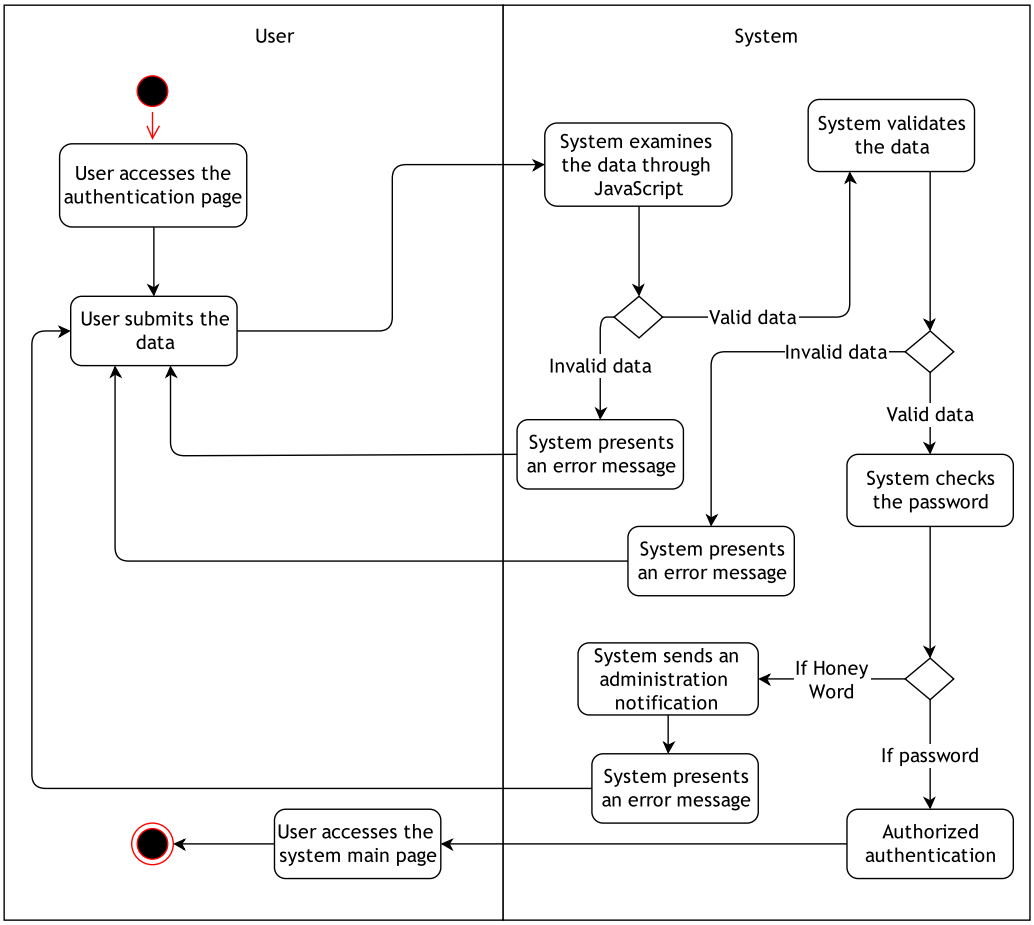


Table 3.4: Diagram and description for the *Manage Projects* related activity.

Use Case name	Manage Projects
Description	1. Initially, the user accesses a list of all his projects in the system. 2. The user is confronted with two possibilities, the access to the team work area, or to the development area (table 3.5). 3. The user accesses the team work area, and adds more elements to the working team. 4. The system checks the data via in the browser (e.g., via JavaSript). 4.1. If a problem is detected, an error message is triggered, and the form to re-enter the data is displayed. 4.2. If successful, the system (server-side) performs data validation also. 5. The system uses protection functions against pre-computed malicious XSS and SQLi codes. 5.1. If the data is invalid, the workflow will return to the beginning. 5.2. If the data is valid, the system calls and sends a notification to the users. 6. A success message is displayed. 7. The user is sent to the project page.
Activity Diagram	

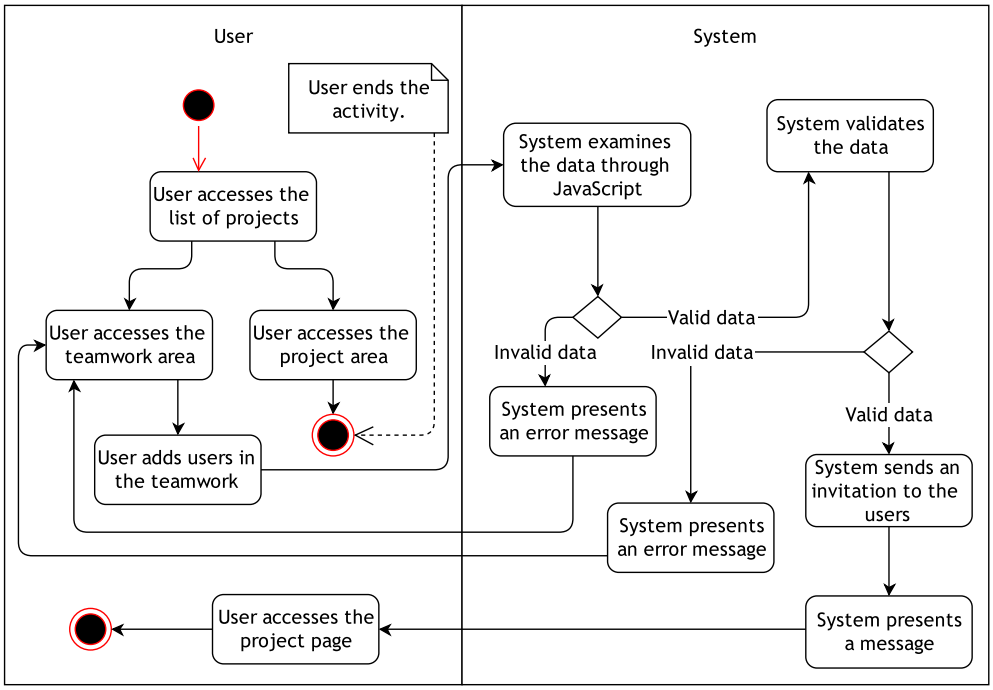
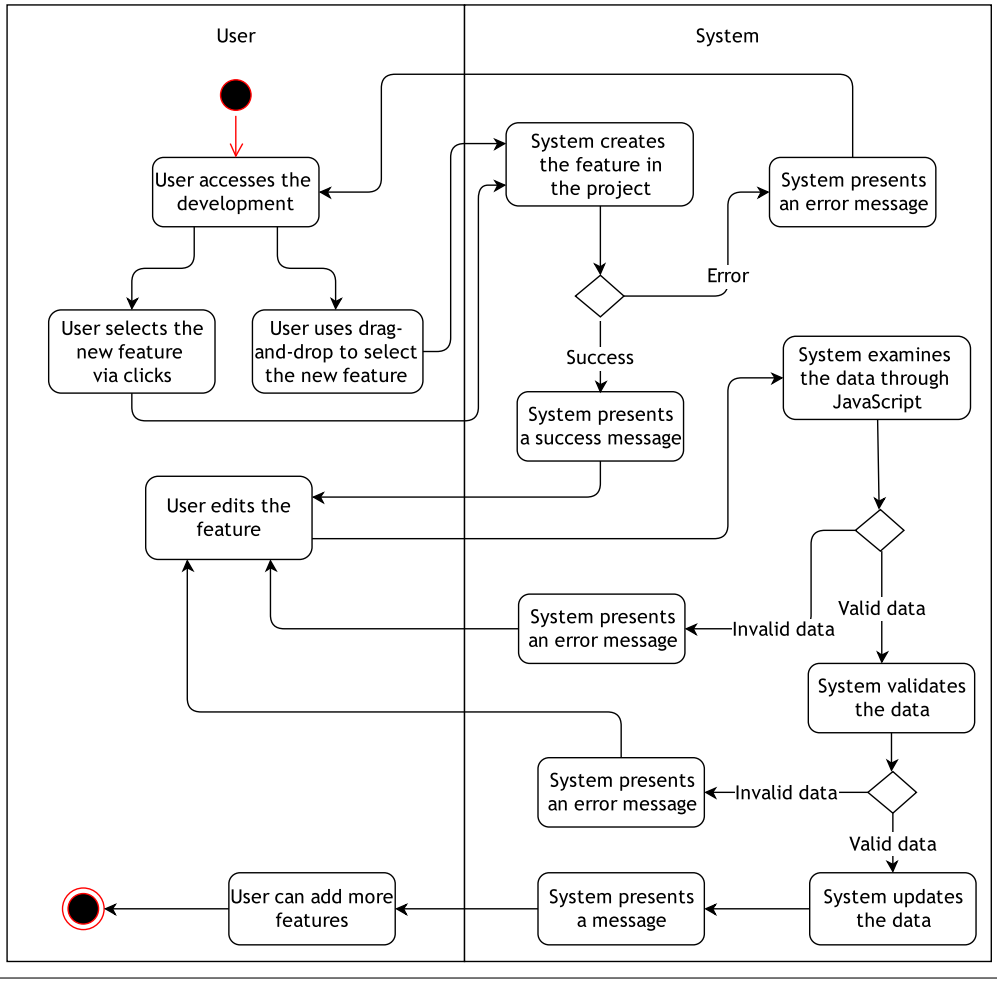


Table 3.5: Diagram and description for the *Select Elements to Layout* activity.

Use Case name	Select Elements to Layout
Description	1. The user accesses the project development page. 2. At this point, the user may proceed with the construction of the new application, by selecting the new feature via drop-down menu or by drag-and-drop of objects. 3. The system add each selected feature to the project. 3.1. If, for some reason, a failure is detected, the changes are not saved. 3.2. Otherwise, a success message is displayed. 4. Subsequently, the user can edit the inserted feature (e.g., add text, images, etc.). 5. The user can then introduce data or manipulate properties. The modifications are submitted to the system so that they can be initially validated in the browser (e.g., via JavaScript). 5.1. If a problem is detected, an error message is triggered, and the form to re-enter the data is displayed. 5.2. If successful, the system (server-side) performs data validation. 6. The system uses protection functions against pre-computed malicious XSS and SQLi codes. 6.1. If the data is invalid, the user will be redirected to the beginning of the process. 6.2. If successful, the system changes the data and saves the corresponding project files, storing some data in the supporting database if needed. 7. Finally, a success message is displayed. 8. The user is redirected to the main development page.
Activity Diagram	



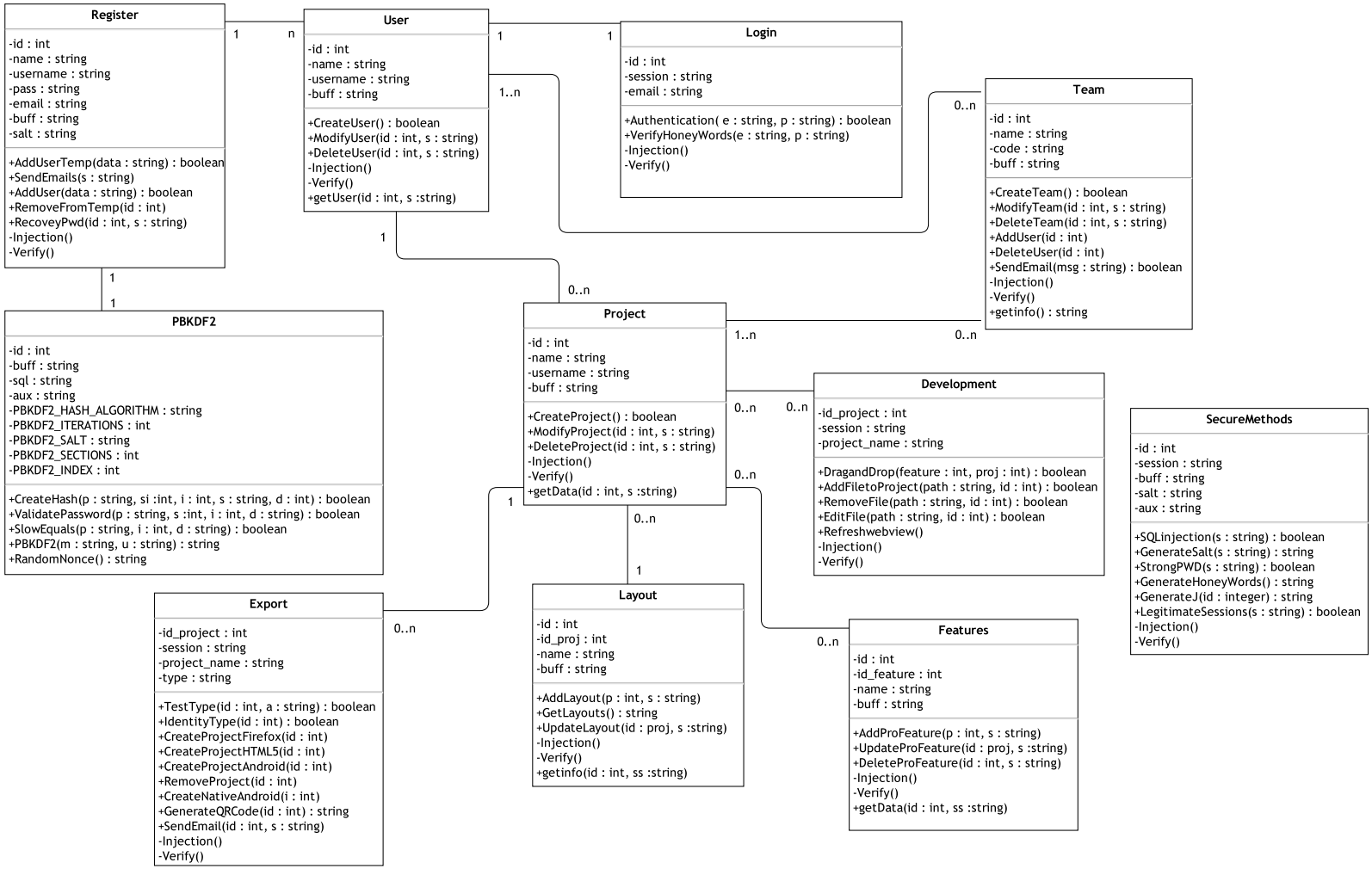


Figure 3.5: Representation of the Classes Diagram of the framework.

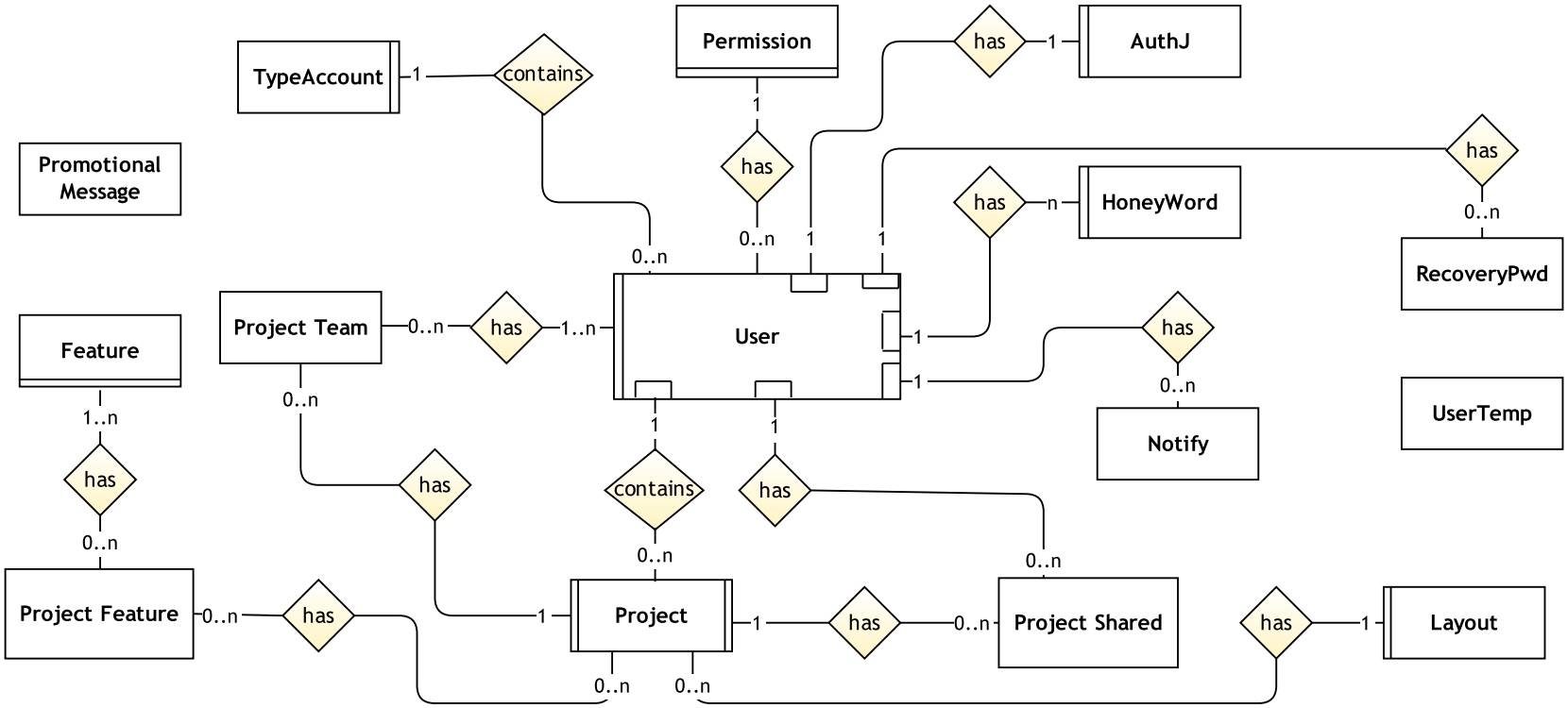


Figure 3.6: Representation of the Entity-relationship Diagram of the system data base.

Chapter 4

Implementation and Security of the Framework

4.1 Introduction

One of the main objectives of this masters programme is the conception of a CPMD. This kind of web frameworks are usually implemented resorting to JavaScript and PHP, integrating a CPT to provide direct access to the native API of the mobile OS. The CPMD developed within the scope of this masters, hereinafter called Expertnear, was completely designed and implemented from scratch, and it does not integrate any external CPT. Instead, it contains its own software to produce both native and non-native mobile applications, thought the number of features available for the native mobile applications is limited.

Expertnear was implemented according to the guidelines provided by the software engineering discussed in the previous chapter. As such, the objective of this chapter is not to thoroughly discuss the entire implementation of the system, but only to present some of the most important details, namely in terms of security and features. Notice that the way to impulse the creation of capable applications is directly connected with the wide range of features that the CPMD provides and to the supported target OSs. Expertnear supports exporting to three different OSs or technologies, namely: (i) HTML5, (ii) Firefox OS and (iii), Android OS. The following sections present an overview on the adopted approach to develop the framework. The approach and organization of the framework is given in section 4.2, while some interesting implementation details will be the subject of discussion of section 4.3. Section 4.4 is then devoted to the discussion of aspects related with the security and performance of the developed system.

4.2 Approach and Organization of the Framework

This section contains a discussion concerning the structure of the developed system. It starts with a description of the model adopted in order to organize all the modules and their communication paths. From a macro perspective, the architecture of the system is the one of the Client-Server and the clients communicate with the server over a computer network using TCP/IP connections. The server component provides a service to one or many clients simultaneously. The clients are always the ones sending the requests for such services. To provide the user-server interaction, a user interface was developed using HTML5, CSS and JavaScript. PHP was the programming language used at the server side to implement most the functionalities provided by Expertnear. Since PHP is object oriented, translating the class diagram into code

was straightforward.

In order to deliver capable mobile applications, the system has to communicate with the native APIs of the respective platform. PHP proved to be a crucial element in this phase also, because it eases the process of interacting with the several SDKs, namely the Android and Firefox SDKs.

The diagram in figure 4.1 depicts an organization for the files that make up the developed CPMD and also the interconnections of the modules or functionalities that they implement. `Indes` lies at the top of the hierarchical layout to denote that this is one of the most important software modules of the platform. It contains the procedures for handling the interaction with the users when they access the main system interface. As this is a fairly complex system, it refers to a wide panoply of procedures.

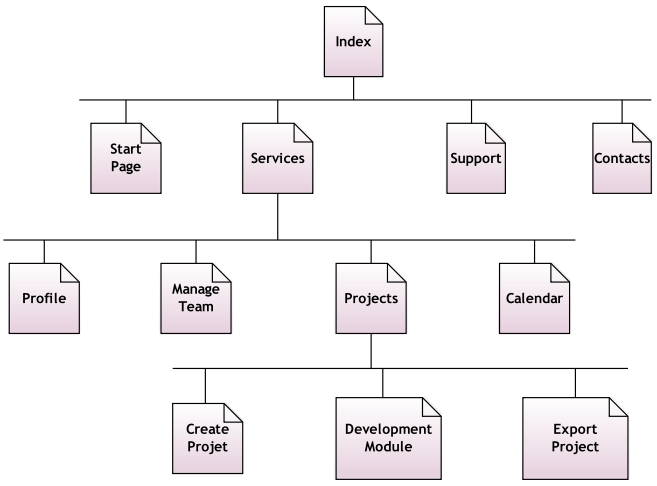


Figure 4.1: Organization of the platform structure.

`Projects` comprises another interesting software modules, mostly because it incorporates a subset of more specific modules related with projects that the users may create in the platform. When a new user registers, the system immediately generates a new user directory, whose purpose is to store all of the mobile application projects created by that users. A sub-directory will be created by each one of those projects, containing at least two additional folders for storing JavaScript and CSS files. An Index page including the remaining HTML pages of the selected features for the application is also included in the folders. Figure 4.2 contains a graphical representation of the aforementioned structure.

If the user desires to export the final application to the HTML format, the package is simply comprised by the files and structure depicted in figure 4.2. In the case the target is the Firefox OS, some extra files are going to be added, containing mainly some configuration information, since HTML5, CSS and JavaScript are native for this mobile OS. For an Android solution, the system will have to create some dependencies and consequently invoke also some specific func-

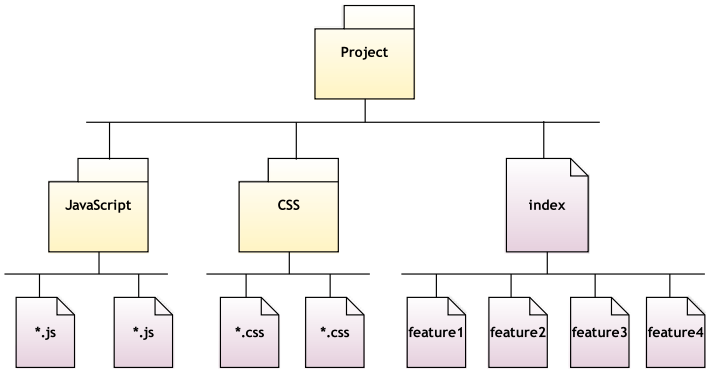


Figure 4.2: Representation of the directory structure of a mobile application project.

tions of its SDK in order to obtain an installable package. To be able to create and compile Android projects, Expertnear requires some environment variables to be set up first, so as to be able to use command line tools for provided with the Android SDK. On a Windows OS, these environment variables can be defined as follows:

`$ set ANDROID_HOME='C:\adt-bundle-windows-x86_64-20130717\sdk\tools'` ,

`$ set ANT_HOME='C:\Program Files (x86)\apache-ant-1.9.3'` , and

`$ set JAVA_HOME='C:\Program Files\Java\jdk1.7.0_51'` .

The `ANDROID_HOME` variable is of utmost importance, because it allows to identify the directory where the Android SDK is installed in the Microsoft Windows 7 OS. The `ANT_HOME` and `JAVA_HOME` variables have similar meanings. The first concerns Apache Ant [Apa], which is a tool used for automating the construction of software. It is similar to `make` [GNU], but it is written in Java and specially designed to be used in projects written also in this language. Finally, the `JAVA_HOME` variable yields the path of to the root containing the Java Development Kit (JDK).

During the construction of a mobile application using the CPMD, some features interactively added via the graphical interface, as for example a news feed, a website, a face detection method, etc. The addition of these features originate new HTML5 pages in the aforementioned project folder. In the end, in order to deliver the project to the Android OS, the platform has to invoke the environment variables listed above to issue the instructions illustrated in listing 4.1.

Listing 4.1: Commands issued during the construction and compilation of the mobile application for the Android OS.

```
1 <?php
2 class Export {
3   function createAndroidPacket(...)
4   {
5     /* include environment variables */
```

```
6       include "android.cfg.php";
7      /* step 1. create the package. */
8       exec($sdk.'android create project --target 1 --name'.$apk .'--path'.$dest
            .' --activity'.  $proj .' --package com.app.'.$pack);
9      /* step 2. creation of files to the solution (ommited). */
10     /* step 3. update the package. */
11      exec($sdk.'android update project --name '.$apk.' --target 1 --path' .
            $dest);
12     /* step 4. build the solution. */
13      exec('ant -buildfile'. $dest .'/build.xml debug');
14    }
15 }
16 ?>
```

The instructions mentioned in the previous listing are critical in the creation of any Android application. All listed commands generate new files. For example, the `android create project` creates a directory in the path given by `$dest` with all files needed for a basic Android application, while the one listed in step 4 creates the archive that can be installed in Android. Notice that, in the step in which the details were omitted, some code is injected in the Java files created with `android create project`. For example, the code for instantiating a *web view* widget may be inserted to enable the loading of the HTML5 files.

Since calls to external programs are being used at this development stage, some software for handling errors returned by this procedure were also included. If an error occurs during the compilation, the system sends feedback to the administrator.

Finally, if the aforementioned commands and related procedures were successful, an Android Package (APK) is generated and consequently sent to the user via e-mail, or made available to download on a link. If such is the case, A QR code encoding that link is also provided for the convenience of the user.

4.3 Implementation Details

A total of 49 different features were implemented from scratch and integrated into the developed CPMD. These features correspond to the individual functionalities that a user of the system can embed into a cross-platform mobile application. This section describes only a sub-set of those features, with the objective of reflecting some of the effort placed in their development. Table 4.1 presents a brief description of 18 different features, along with the name that identify them in the CPMD and with the respective icons. In general, each feature can be understood

Table 4.1: Short description of some of the features provided by the developed CPMD.

Icon	Name	Description
	**Alarm**	This feature allows setting an alarm on the Android OS through a non-native interface.
	**Books**	Provides a set of books available in the cloud.
	**Call Us**	This feature allows to receive calls from the user application.
	**Catalogue**	This features provides the means to make custom catalogs.
	**E-mail Us**	This feature provides the possibility to send e-mails to the seller directly through the application.
	**Events**	This feature presents a listing and description of events. The list is generated from an local SQLite database.
	**Face Detection**	The user can select a particular photo from the gallery and the application displays the number of detected faces in the image.
	**Facebook**	This features allows the user to extract data from a Facebook profile and then create a page with this information organized.
	**Food**	This feature allows the user to create a diet plan.
	**Google+**	Similarly to the Facebook feature, this feature also allows the extraction of data and consequently delivery of a page with the obtained content.
	**Maps**	This feature allows the user to add maps from the Google maps service into the application.
	**Quick News**	This feature allows the user to create a feed of the news included in a SQLite database.
	**Radio**	This feature enables the user to select a radio station and listen to music streaming from the Internet.
	**RSS Feed**	This feature offers the possibility to add Really Simple Syndication (RSS) feeds in the generated application.
	**SMS Bomber**	This feature allows to send unlimited Short Message Services (SMSs) to a given user.
	**Sound Cloud**	The user can include playlists from the Sound Cloud service.
	**Twitter**	This feature allows to import a Twitter feed into the user application.
	**Wall Chat**	This feature allows the users of a given application to communicate via chat.

as a standalone project of the framework. From this point-of-view, it is composed of several projects, which offer the entire range of features. To explain the functioning of some of the features indicated in table 4.1, some of them are described with more detail below.

The *alarm* feature allows the user to configure an alarm on the Android OS. This and all the other features included in the framework provide an interface in HTML5 and JavaScript that allows the communication with the native API. This approach was depicted in figure 2.2 of chapter 2 and it works via the establishment of a bridge between JavaScript and the native API interface. The left side of figure 4.3 contains a screenshot of the user interface for this feature in an Android device. The user interface is composed by two text boxes for configuring the hours and the minutes and a `Call Alarm` button that triggers a procedure for sending the request for the native API which, in turn, triggers a success message on the scree of the phone if correctly executed.

The right side of figure 4.3 contains a screenshot of the Face Detection functionality, which uses a class from the native Android API (the `FaceDetector` class). When the user submits a photo from the gallery to the detection feature, a message with the number of faces identified in the image is displayed on the screen.

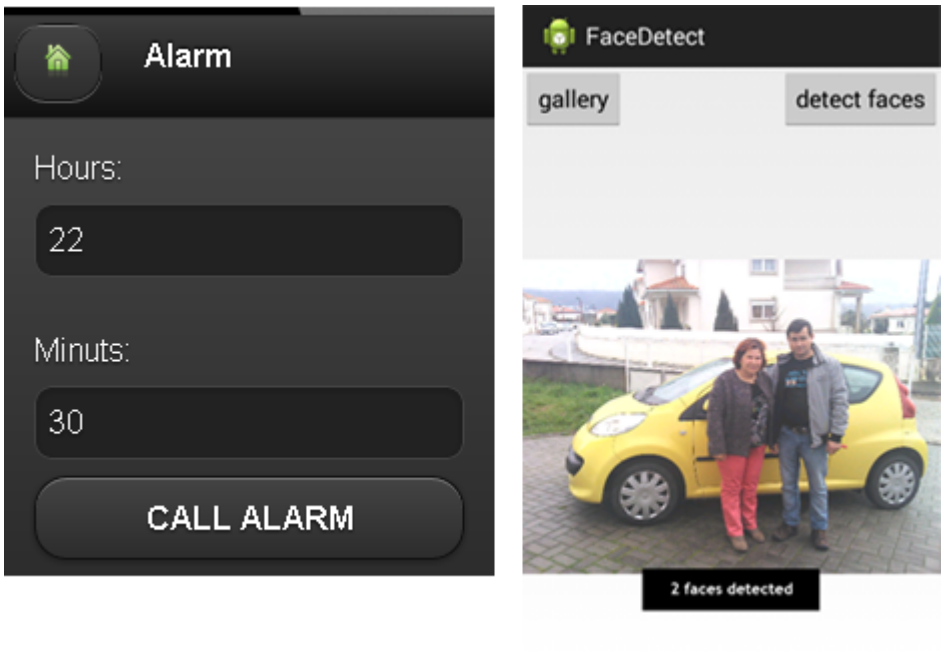


Figure 4.3: Screenshots of the graphical user interface for the alarm and face detection features provided by the developed CPMD.

To implement features such as the ones named *Facebook* and *Google+*, it was necessary to develop web-crawlers. A Web-Crawler is an automated indexer, which extracts content from a particular target in the Internet. Figure 4.4, composed by two screenshots, depicts the settings panel of the Facebook feature (on the left) and the resulting page with the data obtained from the Facebook profile (on the right).

The *Quick News* feature is interesting because it can used to mention the usage of SQLite databases. It also uses a JavaScript interface to communicate with the native API, which invokes the methods to add and delete news.

At this point, it should be mentioned that the easiness of inclusion of new features into the projects was an important detail considered during the development of the framework. It is actually simple to do it, because the user only needs to add a record in the system database and subsequently copy the folder of the new feature to the respective directory of the framework. This folder is copied to new projects using in which the feature is used.

Another important detail concerning the creation of the Android package is related configuration form depicted in figure 4.5. When generating the application, it is necessary to enter the name

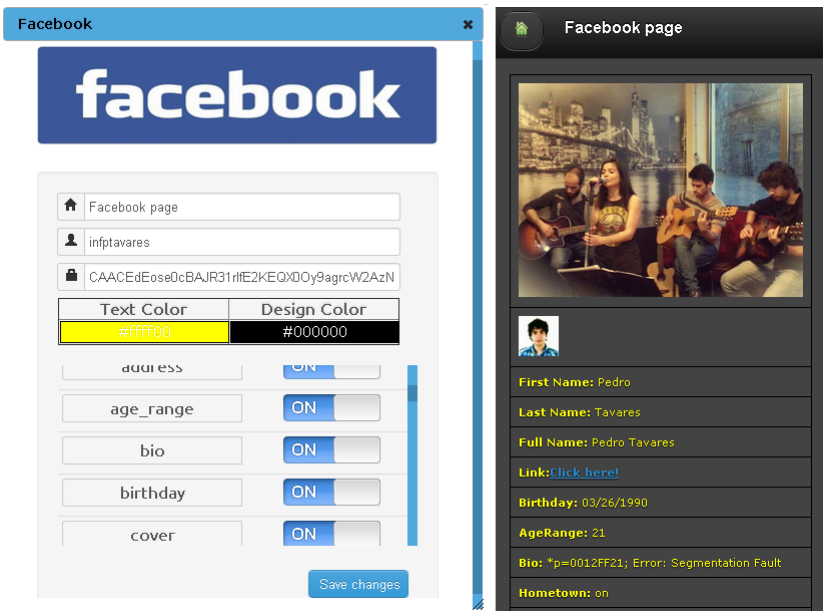


Figure 4.4: Configuration panel of the Facebook feature and a screenshot of the graphical user interface of the non-native feature on the Android OS.

of the APK, the name of the package, the name and the icon that will represent the application in the mobile device.

In order to protect the applications constructed in the developed system, the possibility to define an authentication password after the start of the application was added also. Is the password is set, an hash value is calculated and stored hardcoded in the application. This values is calculated resorting to the Secure Hash Algorithm (SHA)-256 after adding a 128 bit random salt. Every time a user starts the application, the password must be inserted, after which collision between the entered password and the password defined upon creation of the application is computed. If they collide, then the authentication is successful. Otherwise, an error message is displayed.

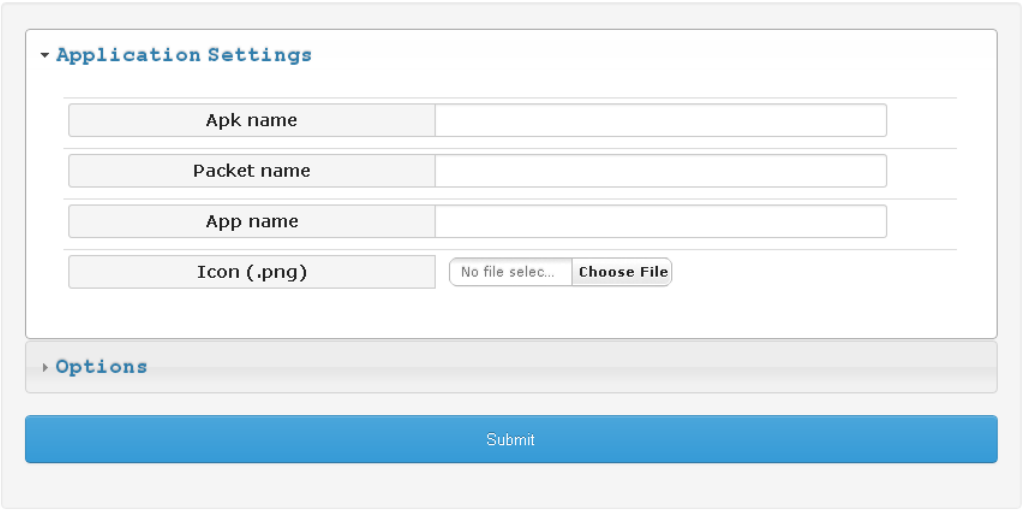


Figure 4.5: Additional settings of the export to the Android OS process.

4.4 Security and Performance Details

Nowadays, web vulnerabilities constitute a hot topic that involves all the systems which use the Internet as their communication infrastructure. An individual with malicious intention does not need to have full knowledge of the web-based systems in order to try to attack them, because they are often very permissive to excerpts of pre-computed codes developed by hackers. In order to protect the developed system, the most widely employed types of attacks to which these systems are typically subject to were taken into account and the respective counter measures were implemented.

Hash Functions and PBKDF2

A cryptographic hash function is any function that can be used to map digital data of arbitrary size to a fixed length digest, with slight differences in input data producing very big differences in output data. They constitute the perfect mechanism to protect the personal password of any user, because the user password is transformed into a piece of meaningless text. To reinforce the randomness component and the difficulty of an attacker to reach an exact collision with the hash value, the data input of the hash function is enhanced with a parameter named salt (a random value). A technique that turns the hash value even stronger (and the process slower), is to use key derivation functions. Password-Based Key Derivation Function 2 (PBKDF2) is an algorithm that applies a pseudorandom function, such as a cryptographic Hash, to the input password or pass-phrase, along with a salt value, and repeats the process many times to produce a derived key, which can then be used as a cryptographic key or just for the purpose of data obfuscation. Figure 4.6 contains a scheme exemplifying this algorithm. By implementing the mechanism

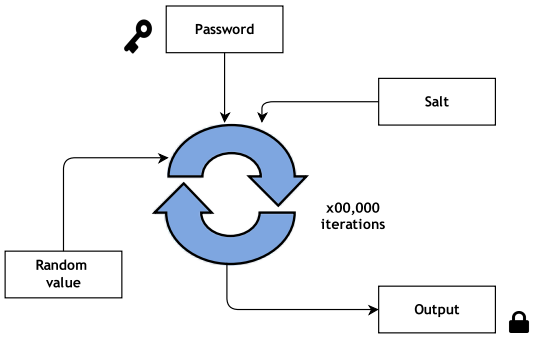


Figure 4.6: Scheme exemplifying the PBKDF2 algorithm for deriving cryptographic keys from passwords.

shown in figure 4.6, it is possible to guarantee that the password of the users cannot be easily recovered from the representation stored in the application or system, unless a really simple password is set. Nonetheless, this mechanism does not prevent attempts to, e.g., compromise the database.

Honey Words

The *Honey Words* mechanism, as already mentioned in chapter 3, namely in the Authentication use case, is a mechanism that prevents the compromise of the system database proposed by Juels and Rivest [Ari13]. Figure 4.7 contains a scheme that may help understand this authentication mechanism. This mechanism focuses on generating several passwords, called *honey words*,

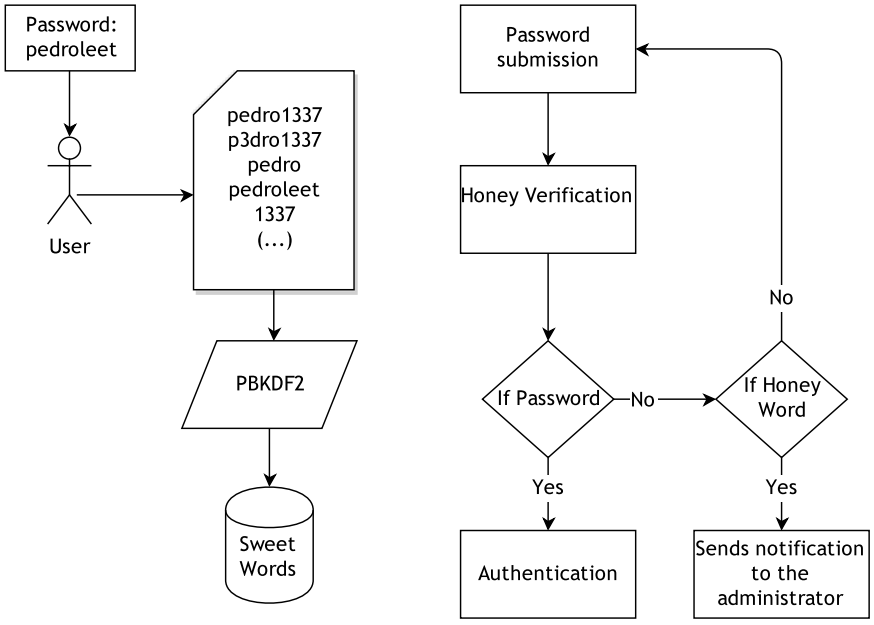


Figure 4.7: Partial representation of the Honey Words mechanism and components.

derived from the password initially defined by the user. Consequently, a hash value for each honey word is generated using the PBKDF2 algorithm, and the results are stored in the table called *Sweet Words*. When the user authenticates to the system, the latter identifies if the hash value of the password collides with the one of the true password or with any of the ones stored in the Sweet Words table. In the former case, the user authenticates to the system. In the latter, the administrator is notified with an alert message.

Sessions and Cookies

In computer science, session hijacking, sometimes also known as cookie hijacking, is the exploitation of a valid computer communication session, sometimes also called session key, to gain unauthorized access to information or services in a computer system. In particular, it is used to refer to the theft of a magic cookie used to authenticate a user to a remote server. It is a very debated subject and attack vector and, as such, it is necessary to take appropriate action. Listing 4.2 shows a snippet of code that handles this particular issue.

Listing 4.2: Snippet regarding session verification.

```
1 if ($_SESSION['123framework123'] != md5($_SERVER['HTTP_USER_AGENT'] . $_SERVER[
      'REMOTE_ADDR'])) {
```

```
2     session_destroy();
3     header('Location: login.php');
4     exit();
5 }
```

After authenticating the user, the system calculated the hash value of the variable `HTTP_USER_AGENT` concatenated with the value of the IP address of the user. Subsequently, this value is stored in the session variable named `123framework123`. If the session is stolen by an attacker and subsequently cloned in his or hers browser, then it is immediately redirected to the login page, because the concatenation of the two aforementioned values does not match the Hash value initially stored in the session variable. As such, this mechanism prevents session theft.

XSS, SQLi and Dangerous Operations

The use of reliable security mechanisms against XSS and SQLi is required in this type of systems. Therefore, all data entries are checked exhaustively with JavaScript functions in the client-side and PHP functions on the server-side. The importance of the server-side checking is due to the fact that a skilled attacker may jump the data check on the client via a local proxy. Using Prepared statements and stored procedures is also important to prevent malicious code to be stored in the database. Using these functions is infeasible to an attacker to pre-compute codes with a format similar to the following one: : `updatefeature.php?id=1' or '1'='1' --`. Since the system relies heavily on databases and on storing information on the server, if this kind of verification was not implemented, the aforementioned types of attack could possibly be used to damage the system after deployment.

On the other hand, any kind of sensitive operations, e.g., delete a project or change the personal password, are preceded by the confirmation of the original personal password. Thus, even if an attacker can get access to the system without the password, dangerous operations are not immediately available, since they need confirmation, and the damage would be limited.

Performance Details

Another important aspect when designing a web-based system is its performance. Total page weight increased by 32% in 2013 to reach a ludicrous 1.7MB and 96 individual Hypertext Transfer Protocol (HTTP) requests per web page. Unfortunately, an overweighted site will adversely affect the user experience and is aggravated by the following factors:

- The larger the download, the slower the experience, and user do not typically like to wait for any site more than a few seconds;
- Mobile web access has increased rapidly to reach almost one in four users. On a typical

3G connection, a 1.7MB page will take almost a minute to appear.

To fight this problem, there are a number of quick fixes which will have an instant effect on system performance. This section mentions some performance techniques used during the development of the system.

Activate GZIP compression -- According to W3Techs.com [W3t14], almost half of all websites forget to enable compression. This is normally a server setting which should be enabled. With this definition, the bundle transmitted between the server and the client suffers a compression and its size is significantly reduced. The way to turn this setting on an Apache server is included in listing 4.3.

Listing 4.3: Code snippet exemplifying the activation of Gzip compression in the Apache HTTP server.

```
1  <ifModule mod_gzip.c>
2  mod_gzip_on Yes
3  mod_gzip_dechunk Yes
4  mod_gzip_item_include file .(html?|txt|css|js|php|pl)$
5  mod_gzip_item_include handler ^cgi-script$
6  mod_gzip_item_include mime ^text/.*
7  mod_gzip_item_include mime ^application/x-javascript.*
8  mod_gzip_item_exclude mime ^image/.*
9  mod_gzip_item_exclude rspheader ^Content-Encoding:.*gzip.*
10 </ifModule>
```

Encourage browser caching -- If the browser can easily cache a file, it will not necessarily need to download it again. Simple solutions include setting an appropriate *Expires header*, *Last-Modified date* or adopting *ETags* in the HTTP header.

Use a Content Delivery Network (CDN) -- Browsers set a limit of between four and eight simultaneous HTTP requests per domain. If the page has 96 assets loaded from a given domain, it may take twelve sets of concurrent requests before all elements are downloaded. (In reality, file sizes differ so it does not happen exactly like that, but the limitation still applies.) Requesting static files from other domains effectively doubles the number of HTTP requests the browser can make. In addition, the user is more likely to have that file pre-cached because it is been used on another site elsewhere. Easy options are JavaScript libraries such as JQuery and font repositories, and it is for this reason that these libraries should be loaded directly.

Concatenate and minify CSS -- Ideally, to increase the performance of loading CSS content, the styles should all be in a single file. During the development phase separate files may be used, but at the end of the development, a single concatenation of the files will increase the system

performance dramatically. To perform this concatenation manually and quickly, for example, on Linux, the following command can be issued:

`cat file1.css file2.css > file.css` .

In the developed system, the aforementioned command is called from PHP before exporting the project.

Concatenate and minify JavaScript -- Again, concatenation may lead to performance gains. On average, each system carries about 18 JavaScript files, and when these are concatenated, important seconds are spared during the page load. Concatenating these files will increase performance, since fewer HTTP requests are also made.

Obfuscation of JavaScript code -- This is an important task when the system possesses sensitive lines in the source code. Since JavaScript run in the client-side, anyone can analyse the source code of the page. In order to hinder this task, the important pieces of code should be obfuscated. To perform this task, there are online tools that can help [Cut14]. In order to understand the differences between normal and obfuscated code, the two following listings were included. Listing 4.4) corresponds to normal JavaScript code and listing 4.5 presents the respective obfuscated version.

Listing 4.4: Non obfuscated JavaScript code.

```
1 xmlhttp.open("POST","ajax_test.php",true);
2 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
3 xmlhttp.send("fname=test&lpwd=123");
```

Listing 4.5: Obfuscated JavaScript code.

```
1 var _0xd332=["\x50\x4F\x53\x54","\x61\x6A\x61\x78\x5F\x74\x65\x73\x74\x2E\x70\
    x68\x70","\x6F\x70\x65\x6E","\x43\x6F\x6E\x74\x65\x6E\x74\x2D\x74\x79\x70\
    x65","\x61\x70\x70\x6C\x69\x63\x61\x74\x69\x6F\x6E\x2F\x78\x2D\x77\x77\x77\
    x2D\x66\x6F\x72\x6D\x2D\x75\x72\x6C\x65\x6E\x63\x6F\x64\x65\x64","\x73\x65\
    x74\x52\x65\x71\x75\x65\x73\x74\x48\x65\x61\x64\x65\x72","\x66\x6E\x61\x6D\
    x65\x3D\x74\x65\x73\x74\x26\x6C\x70\x77\x64\x3D\x31\x32\x33","\x73\x65\x6E\
    x64"];xmlhttp[_0xd332[2]](_0xd332[0],_0xd332[1],true);xmlhttp[_0xd332[5]](
    _0xd332[3],_0xd332[4]);xmlhttp[_0xd332[7]](_0xd332[6]);
```

Resize large images -- Naturally, an image with 3 mega-pixel will produce an image that is too large to display on a web page. Uploading an image with these dimensions through a mobile device connected to a 3G network is not practical and an automated resizing mechanism should

thus be applied.

Consider using Scalable Vector Graphicss (SVGs) -- SVGs contain points, lines, and shapes defined as vectors in Extensible Markup Language (XML). SVGs are ideal for responsive designs since they will scale to any size without loss of quality and the file size is often smaller than a bitmap. For optimum performance, it is essential to use vector graphics.

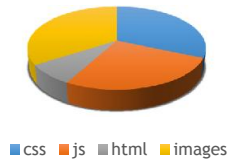
Use data Uniform Resource Identifiers (URIs) -- Data Uniform Resource Identifiers (URIs) encode binary and text assets as if they were external resources. Bitmap images and SVGs can be encoded directly in HTML, JavaScript or, more usefully, in CSS, as shown in listing 4.6.

Listing 4.6: Use of Data URIs.

```
1 .bullet {
2     background-image: url("
          iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD///+l2Z/
          dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4Ug9C9zwz3gVLMDA/
          A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC");
3 }
```

With the application of this technique, the number of HTTP requests is considerably reduced.

To show the impact of the previously mentioned optimization on the performance of the developed system, a simple experiment consisting of measuring the response and loading times of the webpages of the system on a browser before and after application the optimizations was performed. The results were then compiled and are now presented in figures 4.8 and 4.9.



Primed Cache Dataset				Empty Cache Dataset			
12	css	111,15 KB	0,05 s	12	css	141,84 KB	0,09 s
10	js	100,23 KB	0,03 s	10	js	111,15 KB	0,05 s
3	html	8,11 KB	0,02 s	3	html	8,11 KB	0,02 s
13	images	0 KB	0,03 s	13	images	0 KB	0,03 s
Size: 219,51 KB				Size: 261,12 KB			
Time: 0,14 seconds				Time: 0,20 seconds			
Cached responses: 2				Cached responses: 0			
Total resquests: 38				Total resquests: 38			

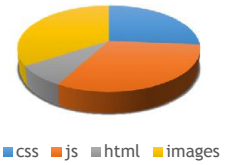


Figure 4.8: Screenshot of the results obtained using Firebug for the accesses to the platform without optimization.

Figure 4.8 summarizes the results concerning the requests and page loading of the homepage of the system without any type of optimization. As can be seen, 12 CSS files and 10 JavaScript files are loaded. The size of the request was 219.51 KB and it took 0.14 seconds to load on localhost. To complete this task, a total number of 38 requests were necessary. After the implementation of the optimizations discussed in this section, the number of CSS files was reduced from 12 to 4 and the JavaScript files from 10 to 3. Additionally, the size of the request, the time of request (decreased from 0.14 to 0.10) and the number of HTTP requests (decreased from 38 to 23) were

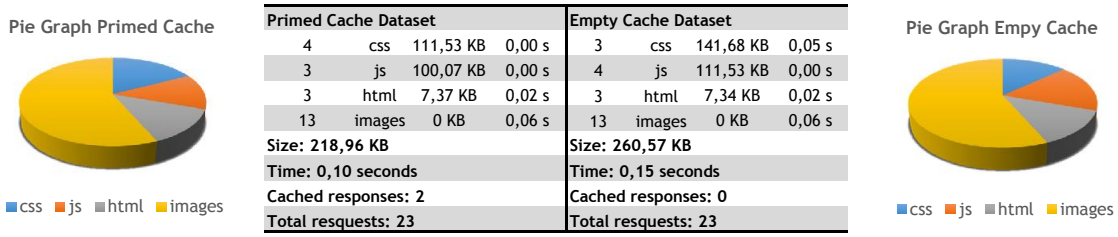


Figure 4.9: Screenshot of the results obtained using Firebug for the accesses to the platform with the optimizations applied.

also diminished, clearly showing that these optimizations make sense.

4.5 Conclusion

This chapter presented some important details on the structure and functionality of the CPMD developed within the scope of this work. Initially, some essential steps that enable the creation of the installation package for the Android OS where described, to then evolve to the description of some of the features that may be included in cross-platform applications created in the CPMD as an excuse to introduce some of the most interesting details and techniques employed. It was shown that some features use JavaScript to communicate with the API of the Android OS, while others run entirely over a webkit view. The simple steps required to use the SDK to generate Android apps from PHP were also enumerated and it was briefly explained how the framework can be extended to include more features.

The central part of the chapter was focused on the implemented security mechanisms, showing that an effort was made to tackle the most severe issues of web-based systems. Security was an important objective of this work all along. As far as the author could assess, this was also the first CPMD implementing the Honey Words database protection mechanism.

Finally, a set of optimizations for making the system faster was described with detail. It was empirically shown that the proposed techniques indeed have a noticeable impact on the performance of the developed CPMD, even though the experiments were conducted in the localhost. The results would be more expressive in a typical client-server scenario.

The next chapter presents a study of the native and non-native applications, exposing their advantages and disadvantages and presents the results of a few performance tests conducted for different scenarios.

Chapter 5

Native and Non Native Implementation

5.1 Introduction

This chapter starts with a brief discussion on some of the disadvantages of cross-platform development. In order to provide the explanation with empirical evidence of the efficiency of native applications, a study comparing native and non-native applications is then presented.

Cross-platform mobile programming is so new and so quickly become popular that it has almost achieved a hype-status. Because of that, there was still no time for the literature on the area to produce much criticism regarding this programming approach. It is thus difficult to produce an objective discussion regarding this matter, since the main sources of information are mostly websites of the frameworks, most of the times containing extravagant advertising language. Most articles on this subject were written between 2008 and 2014 and they do not provide a sufficient or in-depth analysis of the cross-platform frameworks.

5.2 Disadvantages of the Cross-platform Mobile Programming

The cross-platform native frameworks (such as Expertnear) are publicized and marketed with basis on the idea of *code once, run everywhere*. The practical testing of each framework on multiple platforms, with some demo applications, constitutes nowadays the best tool to highlight their omissions, failures and problems. Some evidences of the problems of those frameworks can also be noticed in the discussion and developers forums of the frameworks. If there are many bug reports and questions about the basic functionalities of the framework, the product may be of questionable quality. Both sources were useful *tools* for the task of assessing the advantages and disadvantages of the frameworks and for performing some of the comparisons discussed earlier in this dissertation.

The advertised *code once, run everywhere* idea is not always necessarily true, since some customizations are always needed for each targeted platform. In some cases, minor changes to the UI are enough while, in other cases, some API features are not implemented for all platforms or are functioning differently. PhoneGap which, as mentioned in chapter 2, is the most used *code once, run everywhere* tool in the market is specially good in this aspect.

Apart from what was said, the severe problem haunting this area is that all the cross-platforms (CPTs and CPMDs) are still in their infancy. They still have lot of bugs, undocumented features,

and constant changes to the framework, which may hinder the backward compatibility of the developed application and cause rewriting applications to the new versions of the SDKs. They are unstable, and usually have very limited memory management and debugging possibilities. They do not usually have multitasking or threading capabilities.

5.3 Cross-platform versus Native Applications

Table 5.1 compares native applications with the ones that are cross-platforms and build with frameworks such as PhoneGap and the one developed within the scope of this masters. This comparison is done, for now, via the identification of the advantages and disadvantages of each mobile programming method. With this comparison, it is possible to analyse which choice is the best according to different programming needs.

Table 5.1: Results of the comparison between Cross-platform versus Native applications.

	Native	Cross-Plataform
Advantages	- Library updates - Stable - Code size - Performance - App store compatibility - Native UI - Full native API - Direct comunication with hardware components	- One programming language for all platforms - Common programming language (e.g., JavaScript and HTML5) - Uniform UI design across platforms - Fast development - Multi-platform development
Disadvantages	- Different programming languages, SDKs and tools for each platform -Different UI design - Slow development - Not all are Open Source	- Unstable - Code size - Performance - Lack of multitasking - Debugging - Limited API features - Bugs - Quality - Documentation - UI design in some frameworks

According to table 5.1, native applications are the best choice for applications that require stability (enterprise applications, branded applications), high performance (games), full API features (applications that use device APIs not found in the cross-platform frameworks), and native look and feel for the application (more differences are discussed in section 5.4). It may also be concluded that native mobile application development is worthwhile if the developer has enough financial and time resources available, and the required programming skills for the targeted platforms.

On the other hand, cross-platform development is a good choice if the developers have no native programming skills, the schedule is tight and there the financial resources available for the development are smaller. Cross-platform development constitutes also a good choice if the application uses much web-based data or shares resources with a web site, or if it is a quite

lightweight application that does not require significant hardware resources from the device.

5.4 Comparative Study between Native and Non-Native Applications

One of the things that one needs to decide early on the mobile application development process is the desired method to build and to generate the new applications. The framework proposed in this masters programme allows the creation of non-native applications. However, a module providing the translation from non-native to native code was also implemented, mostly for research purposes. It enabled the comparative study included in this section, which is one of the main contributions of this programme. This comparison is important in the sense that it quantifies the differences between both approaches. The presented study was performed for the following aspects:

- User interface;
- Application response times;
- Performance; and
- Memory.

A single mobile application was designed through the Expertnear framework for the proposed experience. The project then originated the two types of applications: the non-native and the native application for the Android OS. To guarantee that the results obtained for this experiment were meaningful, both applications were tested in two different smartphones, whose specifications are summarized in table 5.2. For more details on the smartphones specifications, refer to [TMN, Sam].

Table 5.2: Characteristics of the smartphones used in the comparative study.

MEO ZTE A7	Samsung Galaxy Fresh
800MHz Processor	1GHz Processor
Internet 3G HSDPA 3.6 Mbps	Internet 3,5G up to 14,4 Mbps
Android OS 2.1	Android OS 4.1
800 MB of Internal Memory	4GB of Internal Memory

The application was designed to include only the features for which a direct translation was available in the framework. Figure 5.1 shows the application design and features used in the scope of this part of the work. The features shown in the figure are briefly described below:

News (`News of World`) - The feature `News` possesses a non-native interface, but its entire work flow is native. It is supported by an SQLite database and the news management is done

via system callbacks.

Figure 5.1: The features included in the applications used in the comparative study between native and non-native applications.

Web Site (`Record`) - The `Web Site` feature comprises a web view that fetches a pre-configured web page in Internet once selected. For this particular feature, it will be important to analyse the response time of native versus non-native applications.

Text (`My Car`) - This feature stores the content defined by the user in a static text (in the native version) or in a HTML web page (in the non-native version).

Send Sms (`SMS Free`) - The button created by this feature spawns an activity for writing and sending an SMS. It will also be important to check the response time of both versions of the application for this feature.

Sobel Filter (`Sobel Filter`) - This feature allows the user to enter, via a dialog, the name of an image to be processed by the sobel filter. It is interesting to study to analyse the response times between the callback and the delivery of the processed image.

Face Detection (`Face Detection`) - the `Face Detection` feature is similar to the previous one, excepts it loads the image via the gallery.

Nowadays, it actually possible to find examples of organizations that develop both native and non-native versions of their applications and one of the major differences between the two is in the look-and-feel. The look of the two applications produced by the framework is depicted in Figure 5.2. The non-native application is depicted in the left side, while the native application is depicted in the right side. Notice that, in terms of the general look-and-feel, there is little difference between the two, suggesting that a consistent user experience is achievable.

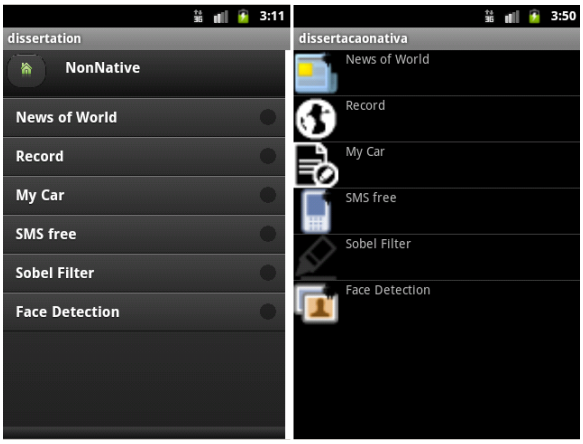


Figure 5.2: Screen captures of the non-native and native applications generated by the framework.

The tables included below and in the appendix C comprise a summary of the results obtained during this experiment for each one of the aforementioned features. Notice that the focus was on the performance, memory consumption and response times of each feature in both versions (non-native and native). Table 5.3 presents the results of the experiment conducted to assess the performance of the applications when starting or exiting. To obtain these results, the time that each application took to start-up after the icon was pressed until the first screen of the application was fully rendered was registered. An analogous procedure was performed for when the application was exiting (in this case for the time it took to render the home screen after closing the application). All experiments were repeated 10 times, and the average and standard deviation values were calculated for the resulting 10 samples. These values were included in the gray lines of the table and used to create the charts in figure 5.3, which emphasize the discrepancy between the performance of the applications. The average times from the non-native application are substantially larger than the ones of the native applications for both operations. This observation is further corroborated by the results included in the remaining tables concerning this study (refer to appendix C), namely table 5.4, which corresponds to the response times of the news related feature. Notice that the values for the standard deviation are also depicted in the charts using intervals represented in the top of the bars with black lines.

The results include in table 5.4 and in the corresponding chart in figure 5.4 emphasize, once again, the poor performance of the non-native application when compared with the native one. This dataset reveals a significant contrast for all of the analyzed operations. In this case, the analysis was more granular, as it measure the response times of starting and exiting the feature from and to the home screen of the application, and also database manipulation operations, namely storing data or deleting it. Though a minor impact on the performance is noticed for the two different smartphones used in the scope of this experiment, the results are consistent for both devices.

This comparative study was performed to all of the aforementioned features of the applications using a procedure that is similar to the one described herein. Some of the tables and figures

Table 5.3: Dataset concerning the overall performance of the applications for cold start-up and exiting. The presented values are in milliseconds (ms).

Performance					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Application Start	Sample 1	470	2560	380	1950
	Sample 2	480	2610	390	1910
	Sample 3	450	2510	380	1840
	Sample 4	470	2940	370	2110
	Sample 5	460	2550	330	1920
	Sample 6	480	2460	370	1890
	Sample 7	470	3100	350	1920
	Sample 8	430	2890	360	1940
	Sample 9	460	2600	340	1990
	Sample 10	450	2490	350	2190
	Average	**462**	**2671**	**362**	**1966**
	Standard deviation	**15,49**	**221,93**	**19,32**	**106,17**
Application Exit	Sample 1	240	680	200	530
	Sample 2	210	590	190	540
	Sample 3	210	600	200	490
	Sample 4	220	610	210	550
	Sample 5	200	690	210	490
	Sample 6	230	630	200	530
	Sample 7	240	660	210	560
	Sample 8	230	650	220	550
	Sample 9	210	620	180	490
	Sample 10	230	670	210	530
	Average	**222**	**640**	**203**	**526**
	Standard deviation	**13,98**	**34,96**	**11,60**	**26,75**

summarizing the analysis were included in appendix C to keep this discussion more concise. Table C.1, included in the appendix, compiles the results obtained for the Web Site feature. It is interesting to notice that, in this particular scenario, the response times for the operation of loading a web page are similar for the native and non-native application. This was expected, as both types load a web-view (which is native) to perform that task.

The results concerning the Text feature, included in table C.2, are useful to expressly conclude about the performance of both implementations. It is a very simple feature, that only exhibits a text message, e.g., `Hello, this is a test!`. The difference between the native and non-native version produces a considerable impact in the final results of this study. The native version uses an individual Android activity, with a text-view component to store the hard coded message. On the other hand, the non-native application uses a HTML5 web page with the message embedded in the source code. The comparatively higher response times depicted in the chart of figure C.2 for the non-native application are directly related with the time it takes to load and render the page in the browser, even if that page is locally stored. The high value of the standard deviation for the non-native applications results also that fact. The time it takes to load the web-view and web page depends on different factors (e.g., if the file was previously loaded or not), making the response times more variable.

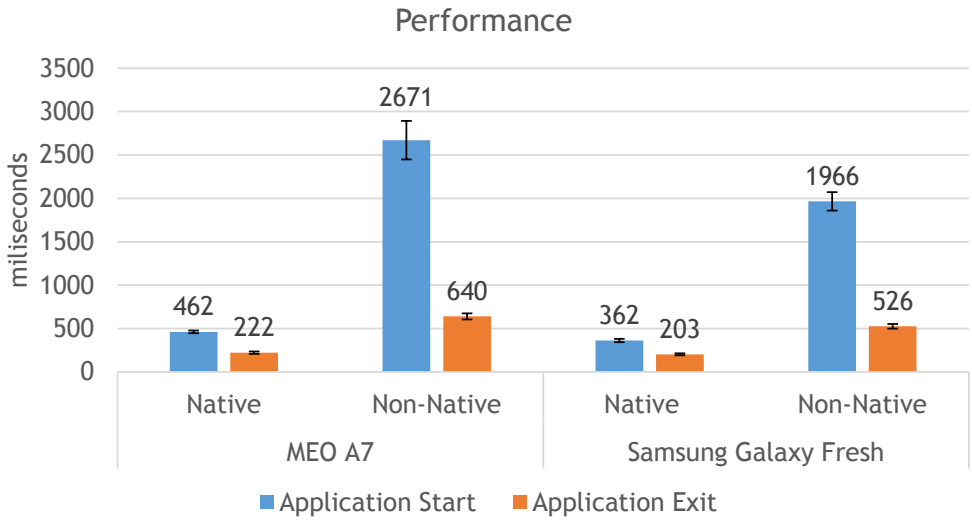


Figure 5.3: Performance of the native and non-native applications when starting or exiting (graphical representation of the data included in table 5.3).

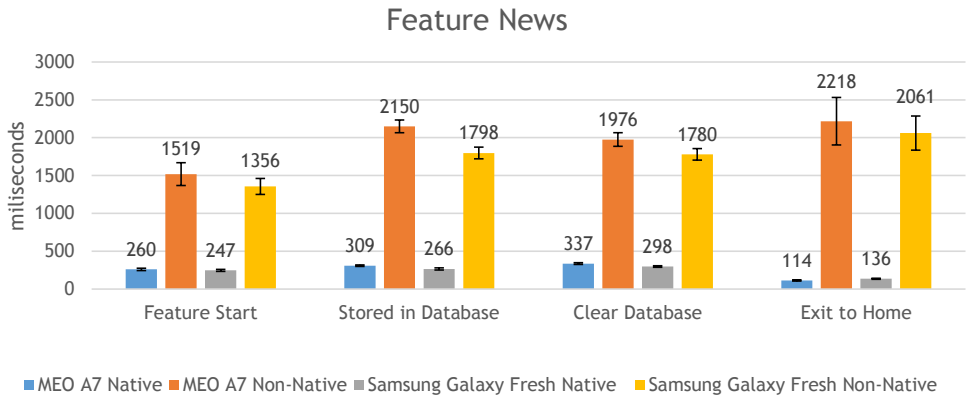


Figure 5.4: Performance of the native and non-native applications for the news related feature and for 4 different operations (graphical representation of the data included in table 5.4).

The final set of results that need to be discussed in this section concern the memory requirements of the native and non-native applications. Table C.6 and figure C.6 summarize the results obtained for three different aspects of this axis, namely the RAM memory footprint and the cache occupancy during execution, the size of the resulting `apk` file and of the `data` and `application` directories. The values for the `Application` and `Data` correspond to the total memory space reserved by the device to store the given application. Once again, the results show the price that is paid in exchange of having an application that works across several platforms. The non-native application consumes significantly more resources the non-native one in any of the analyzed aspects.

5.5 Conclusion

Some advantages and disadvantage of non-native implementations of mobile applications were discussed throughout this chapter. It is evident that this type of programming entails in a set

of specifications and limitations in order to achieve the cross-platform status. Implicitly, it is known that the fact that an application can run in different platforms would prejudice its performance. To better quantify the differences between native and non-native applications, a comparative empirical study was performed resorting to a set of experiments. Basically, a translator for native code was implemented and integrated in the Expertnear framework and two different versions of the same design were generated and submitted to a series of tests. The response times and the memory requirements of both versions were measured for the several features and operations. The results show that the response times of non-native applications are always larger than the ones for native applications, and that the memory requirements of the former are typically more than twice of the requirements of the latter.

Native programming is the best choice to deploy business and commercial applications. Companies developing high performance games should also opt for native implementations. Mark Zuckerberg, the founder of Facebook, once said that *Our Biggest Mistake Was Betting Too Much On HTML5*, revealing that the mobile strategy of Facebook was wrong at that time and needed to be reviewed [Tec12]. CPMDs and CPTs are majestic tools to help the developer or common users to build mobile applications, but is important to have an open the mind and pay attention to the target market and to its requirements. When the application is to serve a very large number of users, as it was the case of Facebook, the company should go through the native programming approach also. On the other hand, a quick prototype can easel achieved with a CPMD framework with a low budget, which is ideal for a small business or a business that is just starting.

Table 5.4: Dataset concerning the response times of the News related feature for four different operations: feature start, exit, storage and retrieval of data from the database. The presented values are in milliseconds (ms).

Feature News					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	270	1380	260	1280
	Sample 2	260	1410	240	1300
	Sample 3	270	1390	240	1570
	Sample 4	240	1530	250	1230
	Sample 5	280	1810	230	1340
	Sample 6	250	1590	250	1400
	Sample 7	270	1450	260	1340
	Sample 8	260	1730	230	1290
	Sample 9	230	1390	270	1500
	Sample 10	270	1510	240	1310
	Average	**260**	**1519**	**247**	**1356**
	Standard deviation	**15,63**	**150,44**	**13,37**	**105,54**
Stored in Database	Sample 1	320	2150	280	1850
	Sample 2	310	2070	290	1890
	Sample 3	320	2210	270	1760
	Sample 4	310	2080	280	1920
	Sample 5	290	2330	250	1750
	Sample 6	310	2050	260	1810
	Sample 7	300	2100	260	1760
	Sample 8	310	2160	250	1650
	Sample 9	320	2140	270	1780
	Sample 10	300	2210	250	1810
	Average	**309**	**2150**	**266**	**1798**
	Standard deviation	**9,94**	**84,06**	**14,30**	**77,29**
Clear Database	Sample 1	340	1920	310	1730
	Sample 2	340	1880	290	1710
	Sample 3	330	1950	300	1810
	Sample 4	340	2030	280	1750
	Sample 5	350	1920	290	1930
	Sample 6	330	1980	290	1790
	Sample 7	350	2030	310	1710
	Sample 8	340	1950	300	1890
	Sample 9	310	2190	310	1750
	Sample 10	340	1910	300	1730
	Average	**337**	**1976**	**298**	**1780**
	Standard deviation	**11,60**	**89,96**	**10,33**	**76,16**
Exit to Home	Sample 1	150	2010	150	1780
	Sample 2	140	1990	140	2130
	Sample 3	140	2230	130	2260
	Sample 4	160	1890	140	1950
	Sample 5	150	2230	130	2140
	Sample 6	130	2150	140	2210
	Sample 7	140	2910	130	1630
	Sample 8	140	2210	140	2220
	Sample 9	150	2590	130	1960
	Sample 10	140	1970	130	2330
	Average	**144**	**2218**	**136**	**2061**
	Standard deviation	**8,43**	**313,86**	**6,99**	**225,51**

Chapter 6

Tests and User Feedback

6.1 Introduction

The final part this work had the objective of obtaining feedback from users during the usage of the developed platform when building an application. The achieve this objective, a study including a questionnaire was planned and put into action. In the developed questionnaire, it was proposed the creation of an application in the framework, to which the answers were related to. This chapter start by presenting the objectives of this feedback study in section 6.2, to then evolve to the presentation of the setup that was created for this study in section 6.3. The results of the questionnaire are them included and discussed in section 6.4.

6.2 Objectives

In the software development process, software testing teams and tests with users constitute important means to improve the Quality of Service (QoS) of many systems. This chapter is not focused directly on the test software paradigm, but rather on obtaining the opinion of users when handling the framework facing the task of creating a mobile/web application. The opinion of the users is collected using a set of questions, orchestrated in such a way that each user is guided through the development phase. The main objectives of the questionnaire can thus be summarized as follows:

- Understand if the respondents already knew these platforms;
- Assess if mobile applications are important in everyday life;
- Evaluate if the developed platform is a vehicle that stimulates the creation of mobile applications;
- Understand if the interaction between users and framework is intuitive;
- Assess if developing and creating applications is an easy task; and
- Understand if the users have the opinion that these frameworks are important.

The questions are ordered in such a way that it becomes easy for the respondent to complete the survey while interacts with the framework. Thought the entire questionnaire is not included

in this dissertation, for the sake of the size of the document, the questions can be found in the charts included below. Nonetheless, it is transcribed here the initial part of the questionnaire designed within the context of this part of the work (notice that the transcribed text is in Portuguese - see below):

Este questionário é de natureza confidencial. O tratamento deste, por sua vez, é efetuado de uma forma global, não sendo sujeito a uma análise individualizada, o que significa que o seu anonimato é respeitado. Este questionário diz respeito a uma plataforma online para criação de aplicações móveis. Por exemplo, poderá construir uma aplicação móvel de forma rápida, eficiente, e para todos os sistemas operativos móveis (e.g., Android e iOS) através do seu browser (navegador web).

Este questionário é constituído, na sua maioria, por questões de escolha múltipla. Durante o seu desenvolvimento, vai-lhe ser pedido que utilize a plataforma de criação de aplicações móveis (endereço disponibilizado a seguir), pelo que pode tomar-lhe entre 15 a 20 minutos do seu tempo. Nas respostas de escolha múltipla, a escala varia de 1 a 5 e denota o seu grau de satisfação relativo ao contexto da questão. Note que a designação de cada grau de satisfação está identificado junto de cada opção.

Endereço da plataforma: `http://expertnear.pt.vu`

The usage of a questionnaire in this part of the work was useful because it enabled collecting information from a large number of users. Nonetheless, this tool has some disadvantages also. For example, it is not typically possible to control the motivation of the respondents, their honesty and skills, which may affect the results of the survey. There were no questions directed to the identification of the respondents, in order to protect their privacy.

6.3 Feedback Study Setup

The questionnaire was elaborated and delivered using Google forms, but in order to hand it out with simultaneous access to the platform, it was necessary to set the latter on a publicly available server. This section is focused precisely on this part of the work.

The delivery of the questionnaire to users was easy, since Internet could be immediately used for that purpose. Most of the respondents were directly contacted via Facebook or e-mail and informed of the Uniform Resource Locator (URL) of the questionnaire. The direct engagement via Facebook was beneficial because it enabled to more effectively convince some of the persons to participate in the survey. There was no specific guidelines for selecting users other than their ability to use computer systems and access the Internet. All questionnaires were

answered remotely. Even though no special selection guidelines were drawn, the final audience was mostly composed by persons with ages between 18 and 32, and most of them were graduate students from different courses. The questionnaire was written in Portuguese due to the targeted audience and all the responders were portuguese also, though the interface to the framework and tutorial are written in English.

To provide remote access to the platform, it was necessary to set the platform in a private server and configure a router to forward requests from the Internet to the internal Local Area Network (LAN). Figure 6.1 depicts a scheme of the communications between the respondent (target users) and the framework. The scheme emphasizes that, for the purpose of this survey,

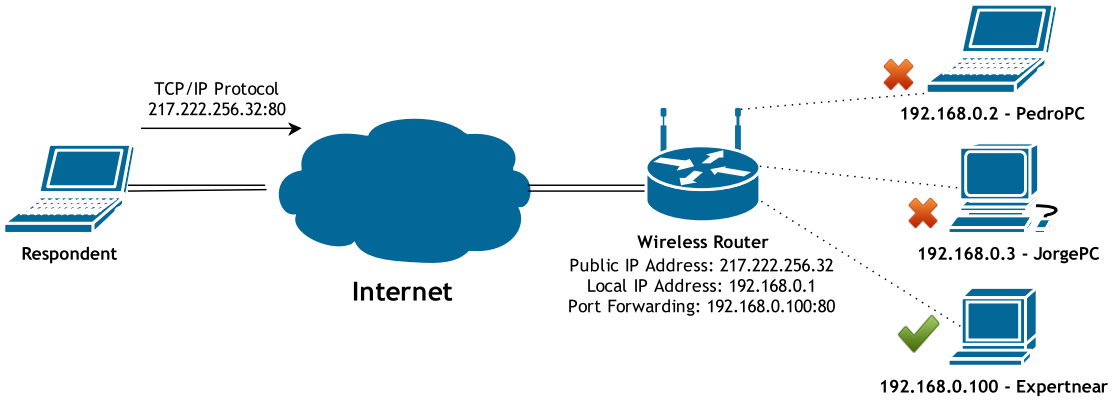


Figure 6.1: Logical representation of the communication infrastructure used in the feedback study.

the framework was configured in a local LAN which, in this case, was a residential/home network. To provide connectivity, it was necessary to configure port forwarding in the residential router, triggered by requests in the external port 80. The rule simply sends the requests to the Expertnear web-server.

6.4 Feedback and Results

The usage of Google forms was beneficial also because they make it simple to collect and analyses the results. In this case, a total number of 39 respondents filled up the questionnaires. The population was constituted by 13 female and 26 male users which, given the context of this study, is a good indicator in terms of gender equity.

Regarding computer literacy, the population is composed by approximately 35% of users that do not currently study or are not connected to the computer science area. From a subjective point of view, this fact favors this study, because the idea is to assess how normal users would deal with the framework. It is believed that it also contributes to a more diverse population.

The results deriving from the questionnaire are presented and described below and in the appendix D. The charts included in this chapter are the ones the author believes to be more inter-

esting for the discussion. From the analysis of the left chart in figure 6.2, it can be conclude

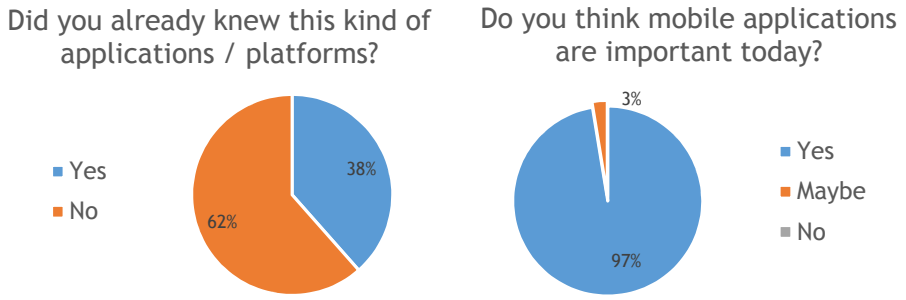


Figure 6.2: Pie charts compiling the results concerning question number 1 and 2 of the questionnaire.

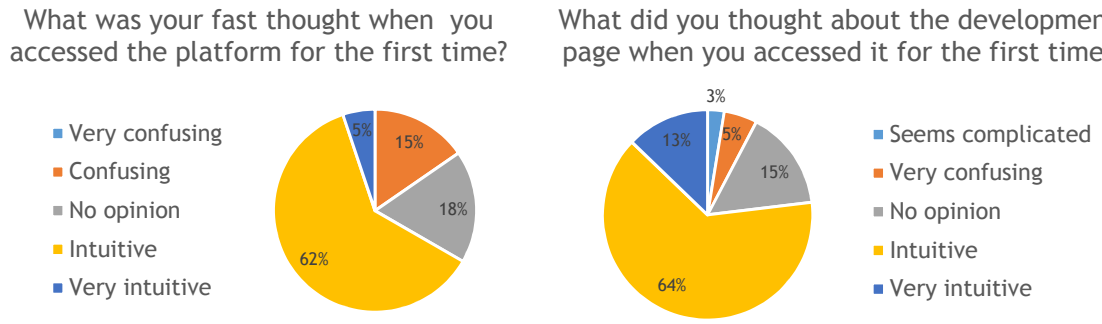


Figure 6.3: Pie charts compiling the results concerning question number 4 and 7 of the questionnaire.

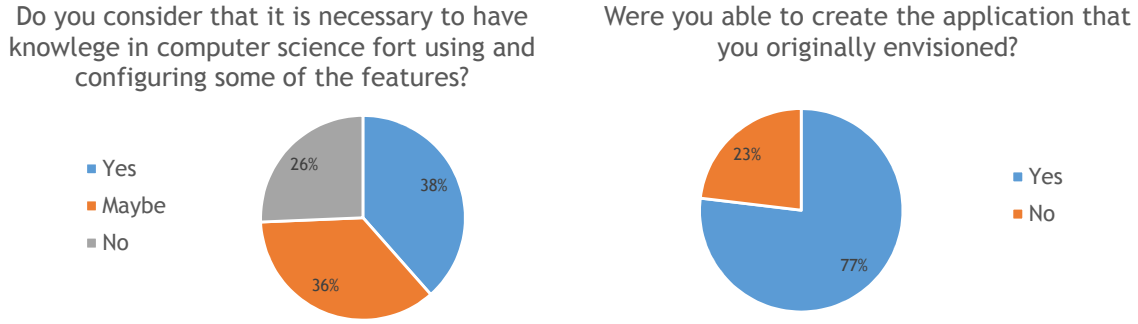


Figure 6.4: Pie charts compiling the results concerning question number 9 and 14 of the questionnaire.

that 62% of the respondents do not know or have no experience with mobile cross-plataform development. This expressive result on this specific matter is probably due to the diversity of the population, because most of them do not have a computer science background and this matter falls within one of its very particular fields. Naturally, the majority (97% of respondents) endorses the idea that mobile devices and applications are essential in the everyday life of any citizen, as emphasized by the results depicted in the right chart of figure 6.2.

Regarding the interaction with framework, many users stated that the platform offers an intuitive interface module (see the left chart of figure 6.3) and a simple development (see the right chart in figure 6.3), which facilitates the construction of a mobile application in an automated way. As for the knowledge in computer science required to manipulate or use certain features

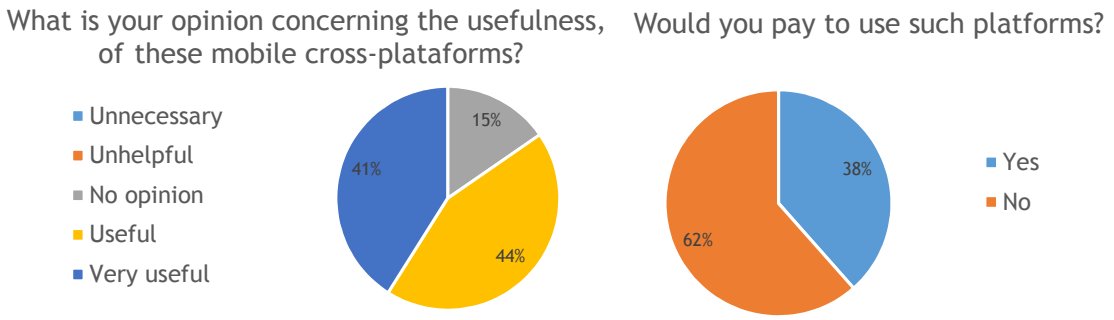


Figure 6.5: Pie charts compiling the results concerning question number 15 and 16 of the questionnaire.

of the framework, summarized in the left chart of figure 6.4, answers show that 38% are of the opinion that it is necessary to have computer knowledge, while 26% agrees that computer knowledge is not necessary for the task to incorporate some functionalities in the new application. The right chart represented in figure 6.4 demonstrates that 77% of the respondents managed to build the application that they had originally envisioned, satisfying their expectations. Finally, the majority of the respondents think that this kind of platforms are very important and useful (see left chart of figure 6.5), but 62% of them are not willing to pay for using it, as evidenced in right chart of figure 6.5. Perhaps a different (corporate oriented) population would give rise to different results on this particular topic.

It should be mentioned that a text box for comments and suggestions was left at the end of the questionnaire. Most of the respondents felt comfortable during the time they were building an application in the framework, stating that in the referred box. Some of the suggestions were related with design and interaction details. For example, it was suggested that some of the buttons were placed differently or the text emphasized, or that some boxes with extra information were added in the tutorial. Though these suggestions were not yet transposed into the current version of the platform, mostly because this survey was performed towards the end of the masters programme, they will be incorporated in the future.

6.5 Conclusion

This chapter described the tasks for preparing a feedback study to evaluate the platform developed along this masters programme and discusses the results of that study. The main tool used in this part of the work was an electronic questionnaire whose questions were aligned with a usage of the platform. 39 respondents participated in the survey, which enabled obtaining some statistically meaningful results. Overall, most of the users of the platform found it to be easy to use and were able to build a simple web/mobile application, even though most of them had no prior experience with mobile-cross platform development. Most users seemed to agree that work flow devised for the platform was adequate, with some suggestions regarding new features and aesthetic changes. While the majority states that they find such platforms useful, they were not willing to pay for such service.

Chapter 7

Conclusions and Future Work

This chapter is divided in two sections. The first section presents the main conclusions of the work developed during this masters programme, while the second section elaborates on some of the points that were not yet addressed in this work and may comprise interesting future research lines.

7.1 Main Conclusions

As briefly motivated in chapter 1, nowadays, mobile devices are considered to be indispensable objects in the daily lives of people in developed countries. Development of applications for mobile devices is thus a very profitable business. Nonetheless, the range of devices and the fragmentation of mobile platforms threatens this industry, since it is hard or even impossible to develop a single application that runs on more than one device or OS. Because of that, CPTs and CPMDs, which are systems that enable building typically non-native applications that run on several platforms, are currently trending. An initial part of this dissertation was dedicated to the subject of cross-platform development. During that part, several already available CPTs and CPMDs were thoroughly reviewed so as to not only emphasize their main advantages and disadvantages, but also to identify the specifications that are required to build an efficient system whose purpose is that of developing cross platform mobile applications. It was noticed that the emergence of new technologies and development methodologies have been contributing and stimulating the recent tendencies in this area. Within this context, Phonegap is nowadays considered to be the best CPMD on the market.

Chapter 2 was devoted to the discussion of the pros and cons of CPMDs and elaborated on their general architecture, providing the context and basis to the work described later on. The comparative analysis performed to the available CPMDs emphasizes that these web-based frameworks are excellent tools for building mobile applications. It is not typically required for the user to have knowledge in the native programming languages of the several platforms, because the functionalities are prepared *a priori* and the interactions with the mobile device are based on HTML5 and JavaScript. Therefore, these frameworks offer a very high level of abstraction to developers, turning them into attractive products if the resulting applications can be exported to the most popular mobile platforms available. Nonetheless, most of these frameworks are paid, being temporarily accessible during a trial period for experimentation. Since these frameworks are more suitable to produce simpler applications under low budget

conditions, the aforementioned fact may hinder their adoption.

The second part of this master dissertation describes the planning and development of a CPMD. Chapter 3 presents its software engineering, elaborating on the requirements analysis and modeling the classes and supporting database. This part of the work enabled the identification of the actors of the system, application scenarios and use cases. The engineering was mostly done resorting to the discussion of UML diagrams, namely some activities diagrams for important features that the system should provide. The supporting database was modelled using an entity-relationship diagram.

The implementation, security and performance details of the developed CPMD were addressed in chapter 4. As the system is part of the web-based applications ecosystem, it may be exposed to a potentially large number of threats, if deployed in a real scenario and, as such, it is of utmost importance to apply good practices in coding and integrate appropriate security mechanisms. Cyber-crime and cyber-terrorism are largely discussed issues in the Information Security (Info Sec) community, with the web-vulnerabilities listed as one of the most critical entry points of attacks. Expertnear is well prepared for the most severe threats known to date, as for example the ones included in the top 10 of the Open Web Application Security Project (OWASP). It includes mechanisms to prevent SQLi and JavaScript code injection, session highjacking and general attacks for compromising the database. It also implements a state-of-the-art mechanism for detecting leaked databases and unauthorized tentatives to access the system called Honey Words. In terms of the performance of the framework, some optimizations were adopted and described in order to reduce time and bandwidth consumption, which are critical also for web-based systems and for improving scalability. Some of those optimizations included reducing the CSS and JavaScript files produced during the creation of an application, along with reducing the number of HTTP request and the bandwidth used to deliver the contents to the browsers. In the localhost, the optimizations alone led to an improvement of 0.14 to 0.10 seconds in loading times. This chapter compiles a large set of useful security and performance guidelines for web-based application developers.

In order to better quantify the difference, in terms of performance and system requirements, between native and non-native applications, a comparative study was performed and discussed in chapter 5. This study was made by conducting a series of experiments with a native and a non-native application, both built in the framework developed within the scope of this masters. It was thus required to implement a translator for native code for some of the features provided by the CPMD. This translator was also implemented from scratch. The experiments were conducted for two different devices and for several features and aspects. They were repeated at least 10 times to achieve statistical significance and assure that a given value was not the product of an execution artifact. The obtained results were very expressive in showing that cross-mobility comes with a significant impact on performance and system requirements. It was left clear that

cross-mobile applications are suitable for when the development budget is low, the expertise in the native language is insufficient or high performance is not a requirement. This comparative study comprises one of the main contributions of this masters.

The next to last chapter of this dissertation discusses an online survey designed with the intention of obtaining user feedback regarding the general topics of cross-mobile development and CPMDs and regarding the usage of the framework developed within the scope of this masters. A total number of 39 participants make up for the population of this survey. The result suggest that Expertnear surpassed the expectations of most users, even thought most them were not really convinced about the usefulness of such frameworks.

Finally, according to the discussion contained in this section, it can be concluded that the objectives delineated for the masters programme were successfully achieved.

7.2 Future Work

One of the most direct lines of future work resulting from this masters concerns the improvement of the developed framework. The survey discussed in chapter 6 emphasized that there is still plenty of room for improvement in terms of user interface, functionalities and work flow. Some of the participants left notes with good suggestions on all of the aforementioned topics.

The documentation of the framework needs to be improved also and the ability to easily add more functionalities and modules needs to be subject to further analysis. Supporting the translation of the features to native code for the iOS platform is also left as future work. Nonetheless, in terms of improvement of the Expertnear CPMD, the most interesting to be added in the short term is the support to publishing the resulting mobile application in the respective software stores, namely Google play, so as to automate the entire process of development. This would leave the framework in a better position to compete with existing solutions.

In chapter 2, some web-frameworks to building mobile applications were described and compared with basis on documentation. It would be interesting, however, to thoroughly analyze all of them by trying them out. This analysis would require idealizing one or more mobile applications, developing them in all the available frameworks, install and test the resulting applications. Several aspects of the CPMD (e.g., the ease of use, the quality and performance of the resulting application and its code, the supported OSs, etc.) would be quantified and monitored during the entire development and testing phase.

Bibliography

[10 13] 10 Excellent Platforms for Building Mobile Apps. http://mashable.com/2013/12/03/build-mobile-apps, December 2013. Accessed April 23, 2014. 17

[Ada11] Adam M. Christ. Mobile application: *Bridging the mobile app gap*. http://www.noblis.org/NewsPublications/Publications/TechnicalPublications/SigmaJournal/Document, March 2011. Accessed March 02, 2014. 16

[Ado] Adobe. Adobe Air official webpage. https://get.adobe.com/br/air. Accessed February 21, 2014. 11, 14

[Ado07] Adobes AIR. Niche or the future of desktop development, http://www.cnet.com/news/adobes-air-niche-or-the-future-of-desktop-development, October 2007. Accessed April 13, 2014. 82

[Ado13a] Adobe. Building and packaging mobile apps in Dreamweaver. http://www.adobe.com/devnet/dreamweaver/articles/dw-phonegap-mobile-app.html, June 2013. Accessed February 24, 2014. 13

[Ado13b] Adobe PhoneGap Build Community. https://build.phonegap.com, 2013. Accessed April 12, 2014. 13

[Apa] Apache. Java JDK Website, http://ant.apache.org. Accessed Jully 18, 2014. 43

[Appa] Apple. iOS. http://www.apple.com/pt/ios. Accessed February 15, 2014. 7

[Appb] Apple. Objective C. https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. Accessed February 21, 2014. 13

[App14] Apple. App Store Sales Top 10 Billion in 2013, http://www.apple.com/pr/library/2014/01/07App-Store-Sales-Top-10-Billion-in-2013.html, Jannuary 2014. Accessed February 24, 2014. 2

[Ari13] Ari Juels, Ronald L. Rivest. Honeywords:*Making Password-Cracking Detectable*, 2013. Accessed August 22, 2014. 31, 33, 34, 49

[Bla14] BlackBerry. BlackBerry OS. http://pt.blackberry.com, 2014. Accessed February 18, 2014. 7

[Bri14] Brighthand webpage. Android and iPhone on market. http://www.brighthand.com/default.asp?newsID=19793&news=Android+iOS+iPhone+Q42012, February 2014. Accessed February 25, 2014. 7

[Bro] Brock Batten and Michael Schneider. Mobilerodie official webpage. http://mobileroadie.com. Accessed February 21, 2014. 8, 14, 16

[Cor14] Corona Labs. Corona SDK official webpage. http://coronalabs.com/products/corona-sdk, 2014. Accessed February 21, 2014. 11, 14

[Cro12] Cross-platform developer tools 2012. *Technical report, Vision Mobile*, February 2012. Accessed March 01, 2014. 9

[Cut14] CuteSoft Components, Inc. JavaScript Obfuscator, http://javascriptobfuscator.com, 2014. Accessed August 21, 2014. 52

[Dam11] Damianos Gavalas and Demetre Economou. Development platforms for mobile applications: *Status and trends*. *Software*, IEEE, January 2011. 28(1):77-86, Accessed March 02, 2014. 16

[Dan10] Daniel Dern. *Spectrum, IEEE*, June 2010. 47(6):14 -15, Accessed March 01, 2014. 8

[Dev14] Developer Economics. Developer Economics Q1 2014: *State of the Developer Nation*. http://www.visionmobile.com/product/developer-economics-q1-2014-state-developer-nation, February 2014. Accessed February 21, 2014. xxi, 11, 12, 13

[Fir] Firefox. Firefox OS. http://www.mozilla.org/en-US/firefox/os. Accessed February 20, 2014. 7

[For10] Forrester. Forrester Research Inc.: *Enterprise And SMB Networks And Telecommunications Survey*, North America And Europe, Q1 2010, 2010. Accessed February 22, 2014. 1

[Gar09] Gartner. Gartner Research: *Predicts 2013 Application Development*, https://www.gartner.com/doc/2256415/predicts--application-development, November 2009. Accessed February 25, 2014. 1

[Gar14] Gartner. Gartner Research: *Gartner Says Worldwide Smartphone Sales*, http://www.gartner.com/newsroom/id/2665715, February 2014. Accessed February 18, 2014. 1

[Git] Git. Git Platform. http://git-scm.com. Accessed February 21, 2014. 84

[GNU] GNU. Make Website, http://www.gnu.org/software/make. Accessed Jully 18, 2014. 43

[Gooa] Goodbarber. Goodbarber official webpage. http://www.goodbarber.com. Accessed February 22, 2014. 8, 16

[Goob] Google. Android OS. http://www.android.com. Accessed February 20, 2014. 7

[HP12] HP. HP web OS. http://www.hpwebos.com/us, 2012. Accessed February 21, 2014. 81

[int] "*Topics for Discussion at the Forthcoming Meeting, Memorandum For: Members and Affiliates of the Intergalactic Computer Network*, Washington, D.C.: Advanced Research Projects Agency. 1

[JQu14a] JQuery. ThemeRoller JQuery Mobile. http://themeroller.jquerymobile.com, 2014. Accessed February 21, 2014. 85

[JQu14b] JQuery webpage. JQuery SDK. http://jquery.com, 2014. Accessed February 21, 2014. 15, 81

[Mic] Microsoft. Windows Phone official webpage. http://www.windowsphone.com. Accessed February 21, 2014. 8

[Mon14] Mono. Mono Touch (Xamarim). http://www.mono-project.com/Main_Page, 2014. Accessed February 21, 2014. 14

[Nok13] Nokia. Cross Platform Mobile Architecture. http://developer.nokia.com/community/wiki/Cross_Platform_Mobile_Architecture, March 2013. Accessed March 11, 2014. xxi, 11

[Phoa] PhoneGap Docs. iOS Platform Guide with Phonegap. http://docs.phonegap.com/en/edge/guide_platforms_ios_index.md.html. Accessed February 21, 2014. 13

[Phob] PhoneGap official webpage. PhoneGap. http://phonegap.com. Accessed February 21, 2014. 8

[Qt] Qt project. Qt Modeling Language, https://qt-project.org/doc/qt-4.7/qdeclarativeintroduction.html. Accessed August 03, 2014. 14

[QT14] QT. QT platform. http://qt-project.org, 2014. Accessed February 21, 2014. 14

[Res13] Research2guidance. Cross Platform Tool Benchmarking 2013. http://www.

research2guidance.com/r2g/Cross-Platform-Tool-Benchmarking-2013.pdf, November 2013. Accessed February 21, 2014. xxi, 12

[S. 12] S. Olson J,Hunter, B.Horgen, and K.Goers. *Professional Cross-Plataform Mobile Development in C#*, 2012. Wrox, 1 edition, Pages 112-114. Accessed February 25, 2014. 9

[Sam] Samsung. Samsung GALAXY Fresh, http://www.samsung.com/pt/consumer/mobile-phone/smartphones/android/GT-S7390RWATPH. Accessed June 22, 2014. 57

[Sam12] Samsung Electronics. BADA OS. http://www.bada.com, 2012. Accessed February 20, 2014. 7

[Sen14] Sencha webpage. Sencha SDK. http://www.sencha.com, 2014. Accessed February 23, 2014. 14, 81

[SQL] SQLite. SQLite webpage. http://www.sqlite.org. Accessed February 23, 2014. 82

[Sym12] Symbian Ltd. Symbian OS. http://www.allaboutsymbian.com, 2012. Accessed February 19, 2014. 7

[Tec12] Techcrunch. Mark Zuckerberg, Non-native application, http://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-betting-too-much-on-html5, September 2012. Accessed June 30, 2014. 62

[The12] TheAppBuilder Ltd. TheAppBuilder official webpage. http://www.theappbuilder.com, 2012. Accessed February 22, 2014. 8, 16

[Tit] Titanium SDK official webpage. Titanium SDK. http://www.appcelerator.com/titanium/titanium-sdk. Accessed February 21, 2014. 8, 13

[TMN] TMN. TMN Smart A7, http://www.fnac.pt/TMN-Smart-A7-Preto-Telemovel-Operador. Accessed June 22, 2014. 57

[Vis12] Vision Mobile. Cross-platform developer tools 2012. http://www.visionmobile.com/rsc/researchreportsVisionMobile, CrossPlatform Developer Tools 2012.pdf, April 2012. Accessed March 05, 2014. 16

[W3t14] W3techs. Gzip Compression., http://w3techs.com/technologies/details/ce-gzipcompression/all/all, 2014. Accessed August 20, 2014. 51

[Wil12] William P. Friese. *Cross-Platform Mobile Development*, October 2012. Pages 1-62,

Accessed February 26, 2014. 10

[Wol12] Wolfram. Wolfram platform. http://www.wolfram.com, 2012. Accessed February 18, 2014. 83

Appendix A

Advantages and Disadvantages of CPTs

This appendix includes the tables with the advantages and disadvantages of each CPTs mentioned in chapter 2.

Table A.1: Advantages and disadvantages of the Phonegap.

Adantages	Disadvantages
-Offers a pure HTML, CSS and JavaScript packaged in a library. -User does not have to understand the native language. This abstraction allows developers to take advantage of these existing skill immediately. -It provides an Eclipse plug-in, but can also be used from other IDEs to build mobile applications. -This follows a plugin architecture, which means that access to native device APIs can be extended in a modular way. -It is open source and free, so there are no licensing costs. -The PhoneGap matured in such a way that offers developers a set features that distinguishes it the great majority. Allows the developers to create applications with the cloud destination without local SDKs. -The developers are free to combine the PhoneGap with another tool like JQuery [JQu14b] or Sencha [Sen14] to produce a better and user friendly UI for the applications. -Supports seven mobile platforms, among them iOS, Android, BlackBerry OS, Windows Phone, HP WebOS [HP12], Symbian and Bada.	-The non existence of support for native UI components design patterns and development tools is a tool failure. -A application can occupy several times the memory size of a native application. This is a critical and severe point for applications generated by this tool. Nevertheless, the devices currently have a good storage capacity, which makes the PhoneGap the mostly tool used by the developers and industry of the mobile development. -The plugin architecture works well if the plugins needed are found timely in Internet, or that your build is completed in a timely manner. -The performance of applications has often been criticized. This is a problem that mitigates non native implementations of mobile applications.

Table A.2: Advantages and disadvantages of Appcelarator.

Adantages	Disadvantages
-The native code output is comparatively quick and smoothly executes on mobile devices. -Appcelarator provides a excellent documentation and potentially supports tablet development. -The setup is easy for new developers. This is an important aspect, due to the competitiveness of these tools in the mobile market. -The use of native UI components is a performance win, and the alloy framework attempts to normalize UI across platforms. -This tool employ JavaScript to normalize code across platforms enabling the leverage existing skills on multiple target platforms. -It provides value-adds such as a Backend as a Services (BaaSs), applications analytics and a marketplace for 3rd party components.	-License and Costs. It provides a community edition of Titanium Mobile free of charge and as open source, although this edition is limited in terms of features. -A large number of API methods are iOS only. -It has restrictive APIs and small set of phones are supported. -The Titaniums spectrum of functionality can be compared to that of PhoneGap. -Compared with the PhoneGap, on your site reports, many questions about problems with the tool are still without any answer.

Table A.3: Advantages and disadvantages of Adobe AIR.

Adantages	Disadvantages
-One of the benefits on using Adobe AIR is the ability to run on a wide array of desktop and mobile devices. -Most of the developers consider the Adobe Air a tool very mature and highly efficient. -It provide a fast execution. The ActionScript 3.0 has a JIT compiler, putting it on a par with Java or .NET technology for raw performance. -An easy installation is also offers by this tool. This provided the runtime has installed successfully, installing AIR applications is likely to be be trouble-free, since all the files go into the application directory. -Adobe Air uses SQLite [SQL], and with this, the applications use a fast local database.	-The access to databases is limited the SQLite and web services. -Proprietary technology. AIR applications depend on Adobes runtime. -Enterprises need to roll out applications over the network in a controlled manner. AIR has no specific support for enterprise deployment. -Adobe AIR do not have threading support. Architect Kevin Lynch says this is a strong candidate for a future release [Ado07]. -In security terms AIR is close to the worst of both worlds, being tightly sandboxed from a developer perspective, but not particularly safe from the users perspective.

Table A.4: Advantages and disadvantages of Sencha.

Adantages	Disadvantages
-Sencha is a tool easier to port to other platforms. -It is can distribute outside the App Store if the application does not require any native APIs. This means callbacks are not made by native API. -The Sencha Touch provides a framework to build applications that work on mobile OS like iOS, Android, BlackBerry, Windows Phone. -It provides native look and like feeling and is recommended in case we are building hybrid mobile application. -One feature that elevates the Sencha compared to the Jquery is the graphical performance. This is fast and fluid on mobile devices better than JQuery Mobile. -This tool has available a pleasurable and complete charting package. -It includes a large library of standard UI components out-of-the-box.	-One of the major problems is the scrolling, still possess some gaps in its operation. -It not have full access to native hardware and OS integration (PhoneGap provides some, but not everything), such as push notifications and local notifications. -Its all JavaScript, which may be more complex than necessary in some instances. -This tool can become complicated if the developer is not proficient in JavaScript. -Since Sencha Touch is not open source, there is the risk of vendor lock in.

Table A.5: Advantages and disadvantages of QT.

Adantages	Disadvantages
-The Qt provides the developer with a substantial set of libraries containing APIs for targeting, networks, animations and more. -This tool have a excellent documentation. This aspect have in mind the comfort and usability supplied to the user. -It is more than a GUI framework. It provides a cross-platform way of doing a lot of stuff that desktop applications often need to do. -Qt is mature and has been vetted by major players, e.g., Wolfram Mathematica [Wol12] has been written on top of Qt since version 7.	-Qts tools are advertised as a "complete tool chain", and QML is a proprietary language specific to Qts stack. Committing to this approach could be seen by many companies as "platform lock-in", given that most businesses are seeking to re-use existing skill sets when adopting a CPT, not fragment skills further. -Originally we listed pricing as a "con", since full commercial support could exceed €4,000 (this includes plugin development, platform development, device modification as well as unlimited support and updates).

Table A.6: Advantages and disadvantages of Corona.

Adantages	Disadvantages
-In general, it takes significantly less code to implement a given functionality with Corona than the Java equivalent. -The Corona simulator runs circles around the Android emulator Google ships with its SDK. It is very easy to change the skin of the Corona simulator to represent a wide array of Android and iOS devices. -Corona have a brilliantly scales graphics. -Setting up the development environment takes just minutes and is much less convoluted than installing Androids environment. -A Corona application is compiled for both iOS and Android systems.	-Debugging Corona applications can be painful. This SDK not have a good IDE or debugger, and the result is troubleshooting even simple issues can be tedious. -Corona use Lua, but their sintax is ugly. Even an experienced and disciplined programmer will have to work very hard not to end up with spaghetti code. -The biggest drawback is that there is free edition of the Corona SDK.

Table A.7: Advantages and disadvantages of Mono.

Adantages	Disadvantages
-MonoTouch still uses Apples UI builder (Interface Builder) in order to build its UIs. Assuming one comes from a C# .NET background, this can be initially uncomfortable. -The MonoTouch is now on par with Xcode with the addition of features such as Integrated Version Support for Git [Git]. -This tool provide full support of C#: Language Integrated Query, delegates, lambdas, events, garbage collection and many other features. -Developing an application in MonoTouch is a much more realistic option as the company behind Mono now has a much more stable future. -The whole .NET framework has been ported and bound to the iOS API.	-Developers still need to develop their applications on an Apple Mac computer (with MAC OS). This is still the single most limiting factor in developing for iOS. -There is still a severe lack of samples, documentation and developer community. -The absence of stability, one of the most infuriating aspects of MonoTouch was that the MonoDevelop IDE would crash periodically or throw stance exceptions. -The biggest disadvantage is that the MonoTouch still is not fully implemented.

Table A.8: Advantages and disadvantages of Jquery Mobile.

Adantages	Disadvantages
-The JQuery is easy and uncomplicated to develop. -It works on wide variety of browsers. -The JQuery community provides a large amount of examples. -It provides the creation of themes using the ThemeRoller [JQu14a], a useful tool for creating themes without writing any line of code. -The layout grids made it easy to create the clients product page, result page, and other custom pages. -The biggest advantage is that JQuery is free, so it is the more common choice of most developers.	-The JQuery Mobile works fine for simple designs where is used a default theme or created an theme using the ThemeRoller. However, when exists something more custom, there are a lot of things that have to be overwritten to obtain the results needed. -The CSS themes styling system has limited options so sites built using this can look similar. -The page transitions and animations do not feel native enough and can be sluggish. -The HTML generated was not as clean as hand coded markup and features many nested *divs* and multiple class names.

Appendix B

Scenarios of Use Cases

This appendix includes the most important Use Case scenarios mentioned in section 3.4 from chapter 3.

Table B.1: Login in the System Use Case and their actions.

Login in the System (Main Scenario)	
Pre-Condition	The user[1] is valid in the system.
Description	1. The Use Case starts when the system requests the data to user (e-mail and password). 2. The system confirms the user data and displays a message ***Welcome to the system***.
Post-Condition	The user can perform the intended operations.

1 refers to the Administrator and normal User.

Table B.2: Use Case Login in the System (side scenario) and their actions.

Login in the system (Side Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case starts when the system requests the data to user (e-mail and password). 2. Then, the user may to use the password recovery. • If user is not valid, and do not have data access, should select the ***New User***. • The user enters the requested data. 3. The user enters your personal data.
Post-Condition	The user can perform the intended operations.

A secure way to control the user authentication and protect the commitment of the system database is the Honey Words mechanism. This is applied for all request authentications from users. Chapter 4 presents this mechanism in major detail. The great important scenarios are exhibited below.

Table B.3: Use Case Manage Users and their actions.

Manage Users (Main Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage users. 2. Then, a list of users is displayed. 3. The user selects a register and can perform two operations: • Editing user data. • Delete the user. 4. This use case also allows the addition of new users in the system.
Post-Condition	The user can perform the intended operations and stores them in the database.

Table B.4: Use Case Manage Users (side scenario) and their actions.

Manage Users (Side Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage users. 2. Then, a list of users is displayed. 3. The user selects a register and select the operation ***Delete User***. 4. Subsequently, a message is displayed to the user, ***Do you want to delete this user?*** 5. If the system detects any malicious action, the session is terminated and display a message ***Error, try again!***. 6. Thereafter, the user must confirm the exclusion of the register, and verifying your e-mail and password for proceed with the process.
Post-Condition	The user needs an authentication in the system.

Table B.5: Use Case Manage Users (side scenario 2) and their actions.

Manage Users (Side Scenario 2)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage users. 2. Then, a list of users is displayed. 3. The user selects a register and may to perform two operations: • Editing user data. • Delete the user. 4. A malicious user (*hacker*), decides to work around the methods of system security. 5. If the system detects any malicious action, the session is terminated and display a message ***Error, try again!***.
Post-Condition	The user must a valid authentication in the system.

Table B.6: Use Case View User Projects and their actions.

View User Projects (Main Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage users. 2. Then, a list of users is displayed. 3. The user selects a register and select the operation ***View Projects Of User***. 4. Consequently, the user can define the status of the project, delete it and view the teamwork.
Post-Condition	The user can perform the intended operations and stores them in the database.

Table B.7: Use Case View User Projects (side scenarios) and their actions.

View User Projects (Side Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage users. 2. Then, a list of users is displayed. 3. The user selects a register and select the operation ***View Projects Of User***. 4. Next, the user can define the status of the project and banish the project. The owner of project will be notified with a message ***Your project was banned!***. 5. The next step is to change the content of project and submit it for a new evaluation.
Post-Condition	Thereafter, the user can advance with the process of development.

Table B.8: Use Case Manage Projects and their actions.

Manage Projects (Main Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to manage projects. 2. Then, a list of projects is displayed. 3. The user selects a register and select the operation ***Edit***, ***Delete***, ***Create Team Work or Export Project***. However, he can creates a new project. 4. At this point, the user can defines the status of the project, delete it and view the teamwork.
Post-Condition	The user can perform the intended operations and stores them in the database.

Table B.9: Use Case Define Layout and their actions.

Define Layout (Main Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to Create a New Project. 2. Then, a list of definitions and types of layouts is displayed. 3. The user selects the desired layout. However, he can change it in other phase. 4. At this point, the user can defines the status of the project and begin their customization.
Post-Condition	The user can perform the intended operations and export them.

Table B.10: Use Case Define Layout (side scenario) and their actions.

Define Layout (Side Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The use case begins when the user selects the option to Create a New Project. 2. Then, a list of definitions and types of layouts is displayed. • If the user desired, can leaves the project as inactive. • Later, if desired, he can continues its inception.
Post-Condition	The user can perform the intended operations and export them.

Table B.11: Use Case Define Layout and their actions.

Select Elements to Layout (Main Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to edit and customize the project. 2. Then, a list of features is displayed. 3. User can choose the most diverse elements and define them for the new application. 4. Finally, the customization of features can be realized.
Post-Condition	The user can perform the intended operations and export them.

Table B.12: Use Case Define Layout and their actions.

Select Elements to Layout (Side Scenario)	
Pre-Condition	The user is valid in the system.
Description	1. The Use Case begins when the user selects the option to edit and customize the project. 2. Then, a list of features is displayed. 3. User can choose the most diverse elements and define them for the new application. • When optimizing features, a verification is performed against malicious pre-computed codes. This prevent the creation of malicious applications for malicious intent. 4. The user must re-enter all the content on the feature again.
Post-Condition	The user can perform the intended operations and export them.

Appendix C

Dataset - Native and Non-Native Implementation

This appendix includes all tables and graphics relative to study accomplished in chapter 5 about the performance, memory and times of response of the non-native and native applications. The listed values are presented in the format of milliseconds (ms).

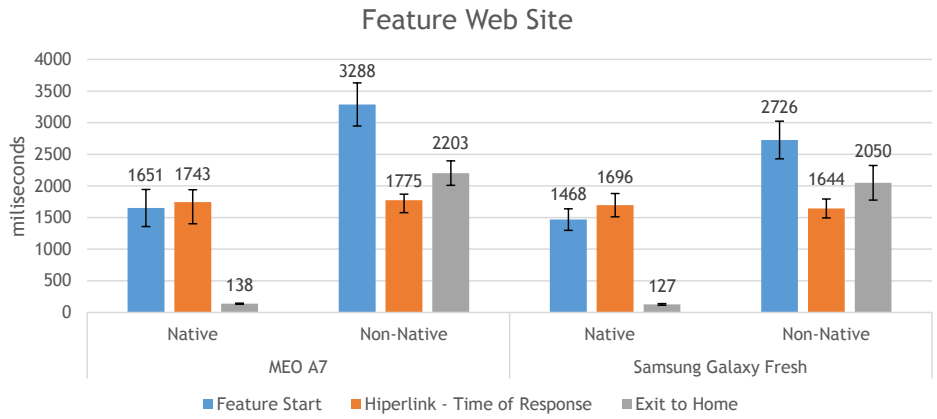


Figure C.1: Performance of the native and non-native applications for the web site related feature and for 3 different operations (graphical representation of the data included in table C.1).

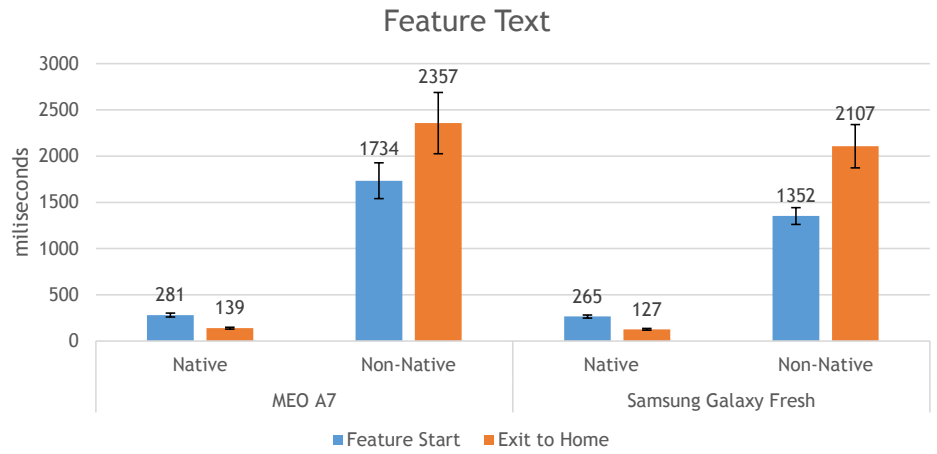


Figure C.2: Performance of the native and non-native applications for the text related feature and for 2 different operations (graphical representation of the data included in table C.2).

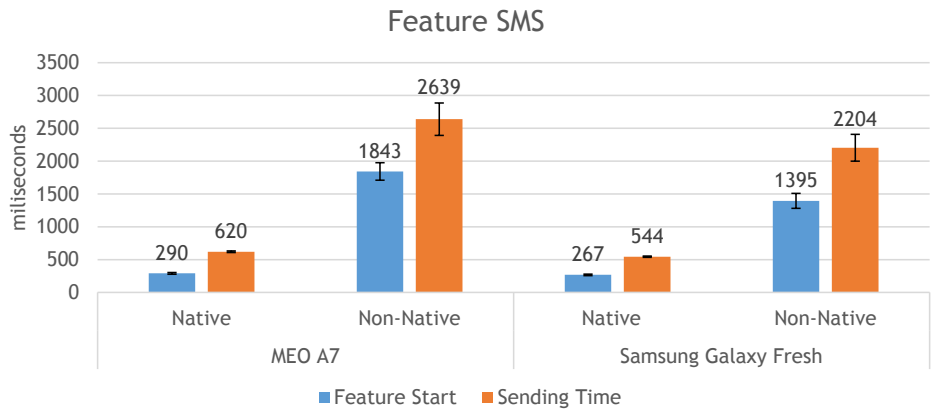


Figure C.3: Performance of the native and non-native applications for the sms related feature and for 2 different operations (graphical representation of the data included in table C.3).

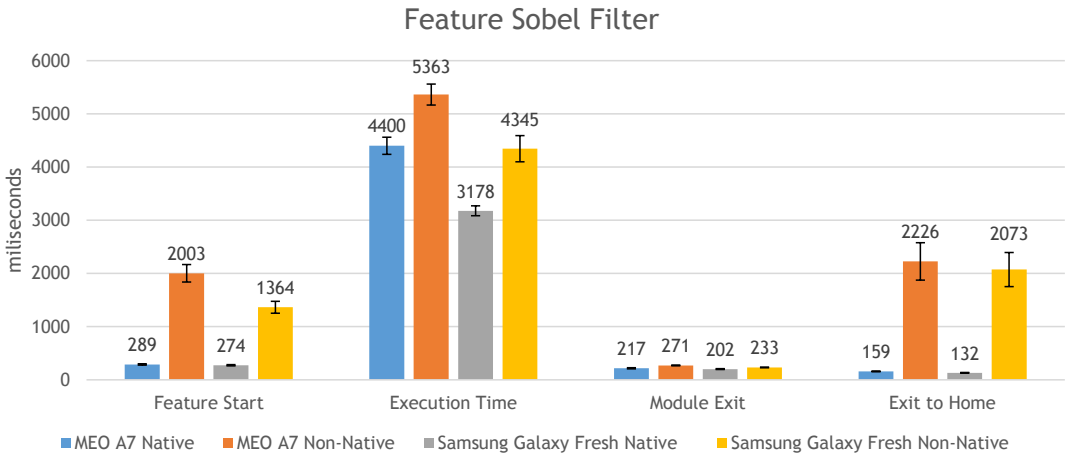


Figure C.4: Performance of the native and non-native applications for the sobel filter related feature and for 4 different operations (graphical representation of the data included in table C.4).

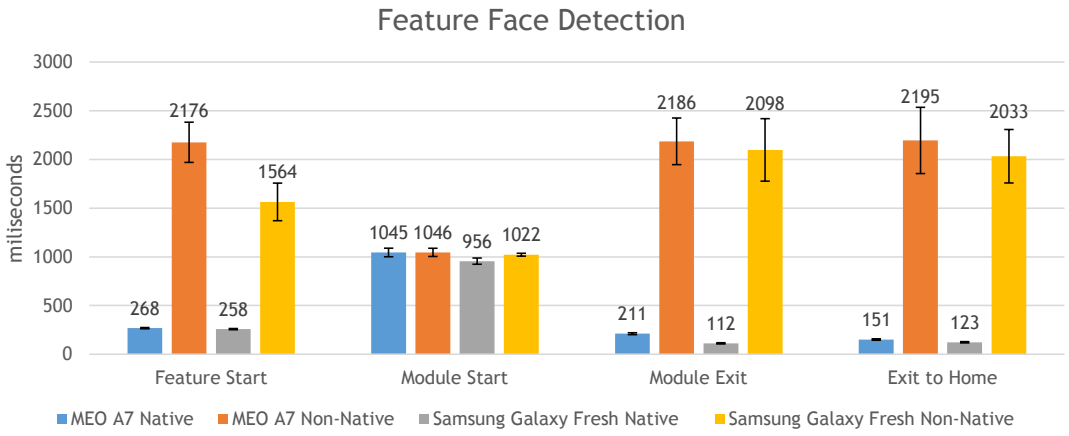


Figure C.5: Performance of the native and non-native applications for the face detection related feature and for 4 different operations (graphical representation of the data included in table C.5).

Table C.1: Dataset concerning the response times of the Web Site related feature for three different operations: feature start, exit and response times of the hyperlink. The presented values are in milliseconds (ms).

Feature Web Site					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	1420	3030	1320	2840
	Sample 2	1520	3650	1290	2230
	Sample 3	1680	2910	1480	2590
	Sample 4	1350	3380	1380	3100
	Sample 5	1520	3150	1310	2400
	Sample 6	2350	2920	1590	2600
	Sample 7	1900	3160	1440	2780
	Sample 8	1720	3560	1860	2600
	Sample 9	1590	3950	1540	3150
	Sample 10	1460	3170	1470	2970
	Average	**1651**	**3288**	**1468**	**2726**
	Standard deviation	**293,39**	**340,91**	**170,28**	**297,70**
Hiperlink - Time of Response	Sample 1	1780	1800	1750	1520
	Sample 2	1410	1640	1730	1640
	Sample 3	1830	1860	1650	1540
	Sample 4	1620	1790	1810	1860
	Sample 5	1710	1640	1980	1710
	Sample 6	1650	1730	1730	1420
	Sample 7	2150	1710	1630	1590
	Sample 8	1690	1850	1250	1620
	Sample 9	1670	1930	1700	1910
	Sample 10	1920	1800	1730	1630
	Average	**1743**	**1775**	**1696**	**1644**
	Standard deviation	**197,15**	**94,90**	**184,22**	**149,76**
Exit to Home	Sample 1	130	2110	120	1710
	Sample 2	150	2090	150	1920
	Sample 3	130	2640	130	1850
	Sample 4	140	2310	120	2420
	Sample 5	140	1970	130	2140
	Sample 6	150	2310	130	2610
	Sample 7	130	2000	140	1950
	Sample 8	150	2140	100	2040
	Sample 9	130	2210	130	1970
	Sample 10	130	2250	120	1890
	Average	**138**	**2203**	**127**	**2050**
	Standard deviation	**9,19**	**193,51**	**13,37**	**273,50**

Table C.2: Dataset concerning the response times of the Text feature related feature for two different operations: feature start and exit. The presented values are in milliseconds (ms).

Feature Text					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	310	1670	290	1300
	Sample 2	290	1660	270	1420
	Sample 3	320	2100	260	1530
	Sample 4	270	1720	270	1350
	Sample 5	280	1560	290	1410
	Sample 6	270	1710	250	1200
	Sample 7	250	1890	250	1350
	Sample 8	280	1450	260	1290
	Sample 9	280	1950	240	1290
	Sample 10	260	1630	270	1380
	Average	**281**	**1734**	**265**	**1352**
	Standard deviation	**21,32**	**193,52**	**16,50**	**90,65**
Exit to Home	Sample 1	150	1990	140	2150
	Sample 2	140	2610	130	2100
	Sample 3	140	2100	130	2610
	Sample 4	120	2430	110	1920
	Sample 5	150	2220	120	2130
	Sample 6	140	3140	130	1880
	Sample 7	140	2420	120	1870
	Sample 8	130	2350	140	2120
	Sample 9	150	2140	130	2360
	Sample 10	130	2170	120	1930
	Average	**139**	**2357**	**127**	**2107**
	Standard deviation	**9,94**	**331,33**	**9,49**	**234,19**

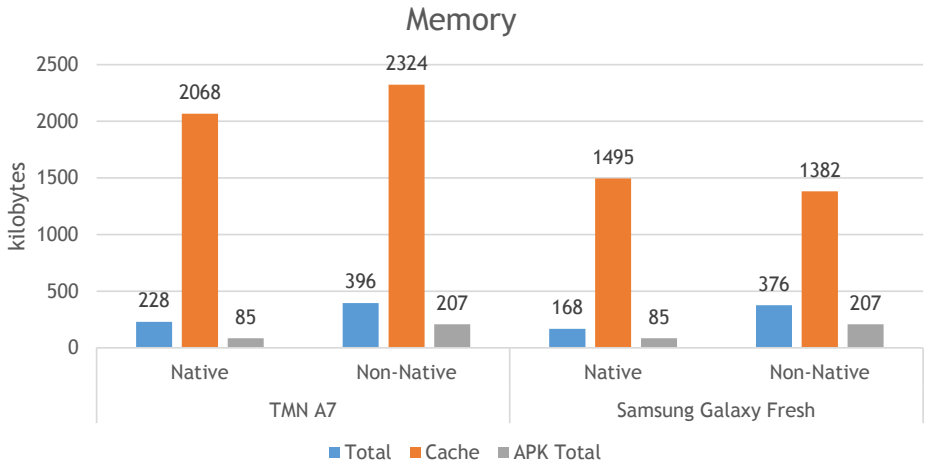


Figure C.6: Bar graphic of the dataset presented in table C.6.

Table C.3: Dataset concerning the response times of the SMS feature related feature for two different operations: feature start and sending time. The presented values are in milliseconds (ms).

Feature SMS					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	280	1770	270	1380
	Sample 2	310	1620	280	1290
	Sample 3	290	1810	280	1310
	Sample 4	280	1790	270	1350
	Sample 5	290	1860	250	1450
	Sample 6	310	2100	280	1290
	Sample 7	280	1960	260	1450
	Sample 8	300	1820	270	1340
	Sample 9	290	1950	250	1670
	Sample 10	270	1750	260	1420
	Average	**290**	**1843**	**267**	**1395**
	Standard deviation	**13,33**	**133,17**	**11,60**	**113,94**
Sending Time	Sample 1	640	2740	550	1950
	Sample 2	610	2420	540	2230
	Sample 3	630	2710	550	2160
	Sample 4	620	2530	530	2020
	Sample 5	630	2540	550	2450
	Sample 6	620	2390	540	2230
	Sample 7	600	2970	540	2640
	Sample 8	610	2500	560	2130
	Sample 9	630	3130	530	2080
	Sample 10	610	2460	550	2150
	Average	**620**	**2639**	**544**	**2204**
	Standard deviation	**12,47**	**246,96**	**9,66**	**204,41**

Table C.4: Dataset concerning the response times of the Sobel Filter feature related feature for four different operations: feature start, exit, execution time and exit module. The presented values are in milliseconds (ms).

Feature Sobel Filter					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	310	1960	290	1400
	Sample 2	290	2100	290	1350
	Sample 3	270	1950	280	1280
	Sample 4	280	1840	260	1400
	Sample 5	300	1790	260	1620
	Sample 6	290	2310	270	1300
	Sample 7	290	2180	260	1190
	Sample 8	300	2100	280	1340
	Sample 9	270	1890	280	1350
	Sample 10	290	1910	270	1410
	Average	**289**	**2003**	**274**	**1364**
	Standard deviation	**12,87**	**164,12**	**11,74**	**111,87**
Execution Time	Sample 1	4520	5380	3100	4310
	Sample 2	4460	5210	3210	4910
	Sample 3	4620	5280	3120	4250
	Sample 4	4190	5360	3250	4300
	Sample 5	4150	5380	3120	4580
	Sample 6	4220	5130	3180	4220
	Sample 7	4480	5410	3320	4120
	Sample 8	4360	5860	3010	4460
	Sample 9	4510	5380	3190	4110
	Sample 10	4490	5240	3280	4190
	Average	**4400**	**5363**	**3178**	**4345**
	Standard deviation	**161,11**	**197,15**	**93,07**	**246,18**
Module Exit	Sample 1	220	280	210	240
	Sample 2	230	270	200	240
	Sample 3	210	260	210	230
	Sample 4	210	270	190	240
	Sample 5	230	270	200	230
	Sample 6	220	260	210	220
	Sample 7	230	270	200	230
	Sample 8	210	280	190	230
	Sample 9	200	270	200	240
	Sample 10	210	280	210	230
	Average	**217**	**271**	**202**	**233**
	Standard deviation	**10,59**	**7,38**	**7,89**	**6,75**
Exit to Home	Sample 1	160	1930	130	1910
	Sample 2	150	2150	140	1850
	Sample 3	160	1810	130	2170
	Sample 4	170	2510	120	2860
	Sample 5	150	2100	130	1800
	Sample 6	160	2490	140	1960
	Sample 7	160	2050	130	2270
	Sample 8	150	1880	140	1960
	Sample 9	170	2930	130	1800
	Sample 10	160	2410	130	2150
	Average	**159**	**2226**	**132**	**2073**
	Standard deviation	**7,38**	**351,64**	**6,32**	**320,49**

Table C.5: Dataset concerning the response times of the Face Detection feature related feature for four different operations: feature start, exit, start module and exit module. The presented values are in milliseconds (ms).

Feature Face Detection					
		Meo A7		Samsung Galaxy Fresh	
		Native	Non-Native	Native	Non-Native
Feature Start	Sample 1	270	2000	250	1420
	Sample 2	260	1950	260	1610
	Sample 3	270	2230	260	1590
	Sample 4	270	2100	270	1630
	Sample 5	280	2650	260	1210
	Sample 6	260	2340	260	1560
	Sample 7	270	2100	250	1970
	Sample 8	260	2250	260	1570
	Sample 9	270	2130	250	1450
	Sample 10	270	2010	260	1630
	Average	**268**	**2176**	**258**	**1564**
	Standard deviation	**6,32**	**206,35**	**6,32**	**193,06**
Module Start	Sample 1	1100	1040	1000	1010
	Sample 2	1080	1030	980	1050
	Sample 3	980	980	890	1040
	Sample 4	1050	1120	950	1010
	Sample 5	1030	1060	960	1010
	Sample 6	1050	1030	970	1020
	Sample 7	1010	1090	950	1010
	Sample 8	1080	1000	990	1030
	Sample 9	980	1030	940	1010
	Sample 10	1090	1080	930	1030
	Average	**1045**	**1046**	**956**	**1022**
	Standard deviation	**44,03**	**42,22**	**32,04**	**14,76**
Module Exit	Sample 1	210	2130	120	1950
	Sample 2	220	1920	120	2580
	Sample 3	210	2300	130	1820
	Sample 4	220	2150	120	1790
	Sample 5	220	2460	110	2670
	Sample 6	210	1970	130	1850
	Sample 7	190	2140	120	1860
	Sample 8	220	2690	130	2140
	Sample 9	210	2130	120	2310
	Sample 10	200	1970	120	2010
	Average	**211**	**2186**	**122**	**2098**
	Standard deviation	**9,94**	**239,41**	**6,32**	**320,79**
Exit to Home	Sample 1	150	1950	120	1560
	Sample 2	150	2410	130	1920
	Sample 3	160	2230	130	2150
	Sample 4	140	1920	120	1850
	Sample 5	150	2310	130	2310
	Sample 6	160	1650	120	1820
	Sample 7	150	1950	120	1950
	Sample 8	140	2790	110	2380
	Sample 9	160	2180	130	2410
	Sample 10	150	2560	120	1980
	Average	**151**	**2195**	**123**	**2033**
	Standard deviation	**7,38**	**340,07**	**6,75**	**274,47**

Table C.6: Dataset relative memory management.

	MEO A7		Samsung Galaxy Fresh	
	Native	Non-Native	Native	Non-Native
Application	112KB	236KB	112KB	236KB
Data	116KB	160KB	56KB	140KB
Total	228KB	396KB	168KB	304KB
Cache	2,02MB	2,27MB	1,46MB	1,35MB
APK Total	85KB	207KB	85KB	207KB

The values listed in this table are defined in Kilobytes (KB) and Megabytes (MB). In order to the exhibit of the graphic the values referent to the cache were converted in Kilobytes.

Appendix D

Results of the Questionnaire

This appendix includes some the results of the questionnaire mentioned and discussed in section 6.4 (chapter 6). The next 8 charts concern some of questions that the other considered as being less important in the context of the feedback study.

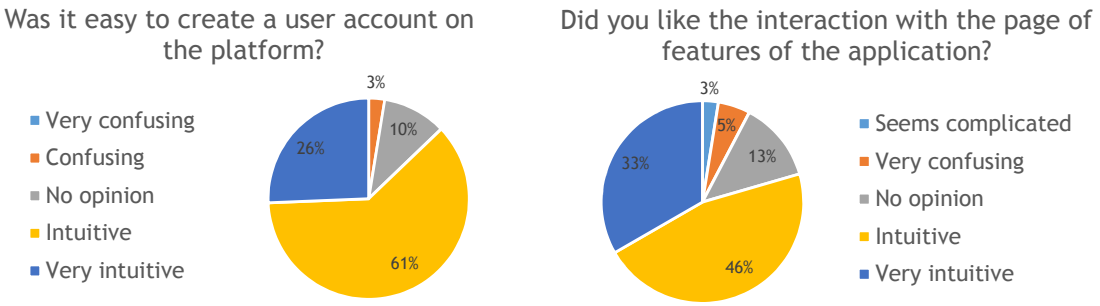


Figure D.1: Pie chart compiling the results concerning question number 3 and 5 of the questionnaire.

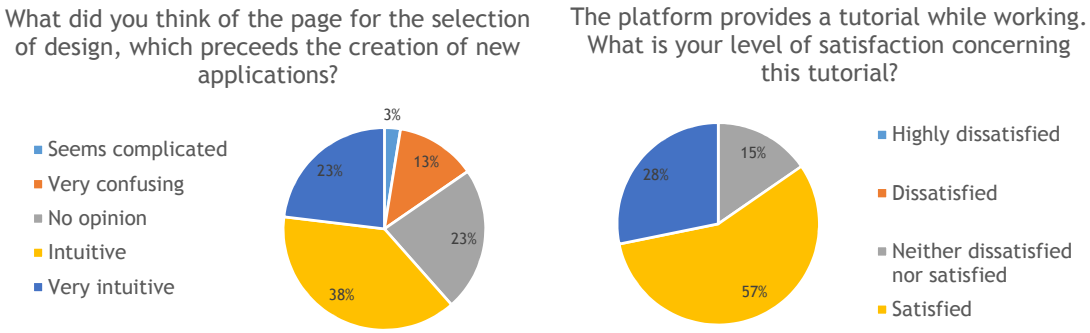


Figure D.2: Pie chart compiling the results concerning question number 6 and 8 of the questionnaire.

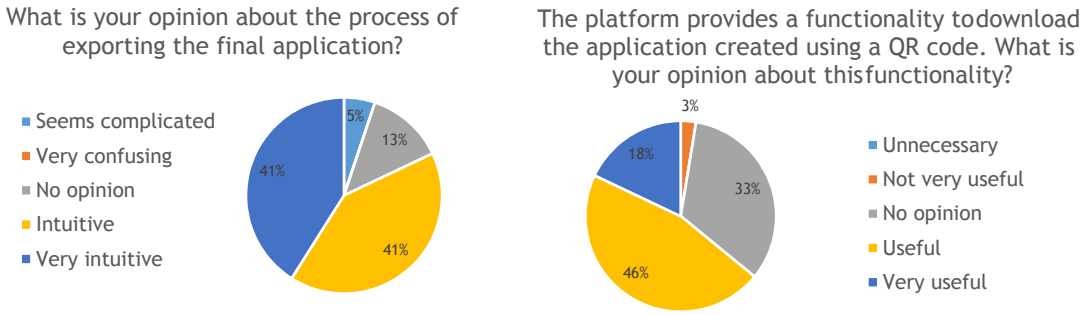


Figure D.3: Pie chart compiling the results concerning question number ten and eleven of the questionnaire.

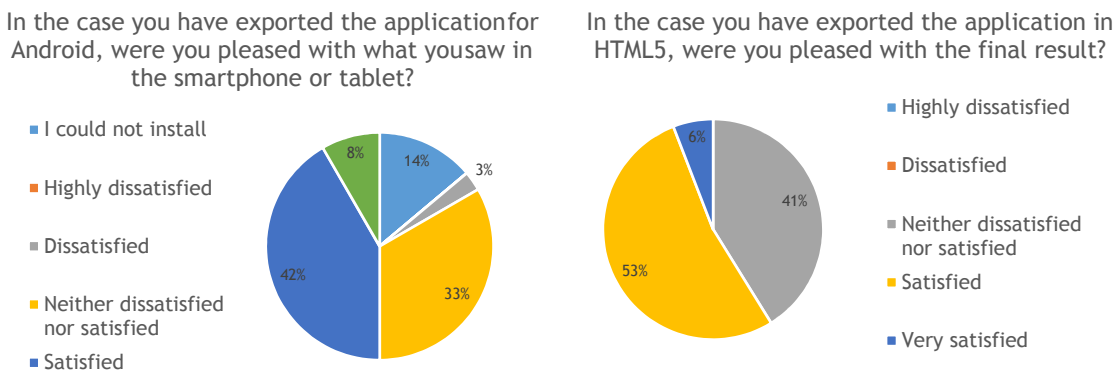


Figure D.4: Pie chart compiling the results concerning question number twelve and thirteen of the questionnaire.