Universidade de Coimbra

Desenvolvimento de Aplicações Móveis recorrendo a frameworks

xpandit

21 de Janeiro de 2017

Autor:

Nuno Miranda
nmiranda@student.dei.uc.pt
DEI, Universidade de Coimbra

Orientadores:

Tiago Cruz
tjcruz@dei.uc.pt
DEI, Universidade de Coimbra

Sérgio Viana
sergio.viana@xpand-it.com
Xpand IT

Resumo

Hoje em dia são cada vez mais as pessoas que têm um dispositivo móvel, seja para navegar na Internet, para consultar o *mail*, para utilizar aplicações ou para a enorme diversidade de coisas que este permite. O dispositivo móvel tem se tornado um objeto tecnológico gradualmente mais essencial no dia a dia de cada um de nós, e existem aplicações que se têm vindo a revelar imprescindíveis simplesmente pela sua utilidade. Para serem produzidas estas tão faladas *"apps"*, existem três grandes sistemas operativos que devem ser considerados para atingir uma maior cota de mercado – são eles o Android, o iOS e o Windows Mobile. Através destas *apps* conseguimos chegar cada vez a mais pessoas das várias faixas etárias.

Posto isto é necessário escolher um dos diversos panoramas de desenvolvimento de aplicações móveis – se optamos por desenvolvimento nativo especificamente para um sistema operativo ou por outro lado uma alternativa *cross-platform*, onde desenvolvemos aplicações para os diversos sistemas operativos atingindo, consequentemente, um maior número de utilizadores.

O objetivo desta tese é o desenvolvimento de uma aplicação mobile para uso interno dos colaboradores da empresa Xpand IT, com o intuito de facilitar as suas tarefas do dia-a-dia, recorrendo a *frameworks* – mais especificamente utilizando Xamarin – e analisando também as restantes propostas de desenvolvimento multi-plataforma que o mercado disponibiliza.

Palavras-chave: cross-platform, xamarin, iOS, android, windows, phone, mobile, software, store.

Conteúdo

Lista de Figuras

Acrónimos

API Interface de Programação de Aplicações

App Aplicação móvel

ARM Advanced Risc Machine

CSS Cascading Style Sheets

GPS Global Positioning System

HTML HyperText Markup Language

IDE Integrated Development Environment

IT Information Technology

LINQ Language-Integrated Query

MVC Model-View-Controller

MVVM Model-View-ViewModel

REST Representational State Transfer

SDK Software development kit

UI User Interface

WORA Write Once Run Anywhere

XML eXtensible Markup Language

Capítulo 1

Introdução

Os dispositivos móveis estão cada vez mais presentes nas nossas vidas. Falamos de pequenos computadores que têm mais e mais funcionalidades e poder de processamento, desde *smartphones* a *wearables*, que nos permitem facilitar cada vez mais a nossa vida. Os *smartphones*, em particular, são um equipamento considerado de crucial importância. Atualmente existem três sistemas operativos que lideram o mercado dos *smartphones*: o Android (pertencente à Google), o iOS (Apple) e o Windows Mobile (Microsoft). Cada uma destas empresas segue uma estratégia bem definida para abordar a mobilidade.

O desenvolvimento de aplicações tem aumentado significativamente e isso pode ser comprovado pela existência de apps de todo género, desde aplicações simples de lazer a aplicações empresariais.

Atualmente existem *smartphones* para todos os gostos (diferente capacidade de processamento, dimensão, resolução, sistema operativo, memória, entre outros). A heterogeneidade de dispositivos torna o desenvolvimento para estes equipamentos numa tarefa complexa, exigindo diferentes tecnologias dedicadas para o desenvolvimento de cada plataforma. Para desenvolver aplicações para as três plataformas referidas anteriormente temos uma grande diversidade de recursos para o fazer.

A forma *standard* é o desenvolvimento de aplicações nativas para cada uma das plataformas, de forma independente, usando as tecnologias que a própria marca disponibiliza (Java usando Android Studio para Android, Objective-C ou Swift usando o Xcode para iOS, e C# usando Visual Studio para desenvolver para as plataformas Windows).

Outra alternativa é o desenvolvimento *cross-platform*. Havendo um enorme

crescimento na utilização dos *smartphones*, existe uma enorme necessidade de rentabilizar e agilizar o desenvolvimento de aplicações nas três plataformas lideres de mercado, sem com isto comprometer a qualidade das aplicações desenvolvidas. Nos últimos anos têm surgido várias abordagens de empresas distintas, focadas em disponibilizar tecnologia que ajude na resolução deste problema, como é o caso do Phonegap ou mesmo da portuguesa Outsystems. No entanto, a maior parte destas abordagens consiste na criação de apps utilizando tecnologias Web que depois são encapsuladas em *containers*, uma particularidade que muitas vezes obriga a compromissos em termos de usabilidade, performance e acesso às APIs. O cenário ideal seria poder utilizar tecnologias familiares sem ter que forçosamente escolher prejudicar algumas destas componentes. É precisamente isso que promete a Xamarin[1], empresa criada em Maio de 2011 e entretanto adquirida pela Microsoft [7] como parte da sua estratégia de complementar as ferramentas para *developers*. Em Xamarin é possível desenvolver aplicações para Android, iOS e Microsoft tendo como base a mesma linguagem de programação (C#), obtendo o mesmo desempenho e qualidade de uma aplicação desenvolvida de forma nativa. Tudo isto é possível recorrendo a *frameworks* .NET (nomeadamente MVVMCross[2]) que mantêm a sustentabilidade do sistema. Para além disto, temos acesso a 100% das APIs das plataformas respetivas - sem limites - usando C#, o que possibilita a implementação de qualquer funcionalidade e experiência de utilizador.

A Xamarin está focada na resolução de vários problemas relacionados com o desenvolvimento mobile multi-plataforma, nomeadamente na dificuldade em garantir a qualidade das apps, num mundo onde cada vez há mais dispositivos diferentes. Porque não faz sentido desenvolver aplicações se estas apresentarem problemas ao serem usados pelos utilizadores finais, a Xamarin criou o Xamarin Test Cloud[3] e o Xamarin Insights[4]. A primeira ferramenta providencia uma forma de realizar testes em milhares de diferentes dispositivos físicos, uma *device cloud* acessível a partir de qualquer ponto no globo. Isto permite garantir que a aplicação execute de forma correta e eficiente numa enorme variedade de dispositivos com um custo mínimo [9]. A segunda ferramenta, Xamarin Insights, permite a monitorização das aplicações tanto em termos operacionais (*bugs* e *crashes*) como em termos de utilização efetiva, e produz um relatório respondendo a diversas perguntas como: "A minha aplicação *crashou*?", "Qual o *crash*?", "O que o provocou e como o replicar?".

[1] `https://www.xamarin.com/` (acedido a Fevereiro de 2016)

[2] A framework mais utilizada por desenvolvedores de Xamarin. URL: `http://mvvmcross.com/` (acedido a Fevereiro 2016)

[3] Xamarin Test Cloud: `https://www.xamarin.com/test-cloud` (acedido a Fevereiro 2016)

[4] Xamarin Insights: `https://www.xamarin.com/insights` (acedido a Fevereiro 2016)

Esta tese visa o desenvolvimento de uma aplicação mobile utilizando as diversas ferramentas disponibilizadas pela Xamarin.

1.1 Definição funcional

O objetivo consiste em desenvolver uma aplicação *mobile* corporativa para a empresa Xpand IT, com funcionalidades em diferentes áreas relevantes para uso interno na empresa (a aplicação destina-se apenas aos colaboradores da empresa, sendo que qualquer outra pessoa não terá credenciais para aceder a esta). A aplicação irá ser desenvolvida para a plataforma Android[5] e para a plataforma UWP[6]. Desta forma pretende-se simplificar algumas das tarefas do dia-a-dia dentro de empresa, tais como acesso aos portais internos, contactos da empresa e de cada um dos colaboradores da empresa, morada de cada um dos escritórios, entre outras funcionalidades que irão ser descritas em detalhe mais à frente na secção 7. Foi proposto que para o desenvolvimento desta aplicação *mobile* fosse utilizada uma abordagem multi plataforma (ver secção 2.3) visto que irá ser desenvolvida não só para as plataformas Android e UWP mas também para iOS (apesar desta plataforma estar fora do âmbito deste estágio).

1.2 Tecnologias a utilizar

Como já foi referido anteriormente para fazer o desenvolvimento *cross-platform* irá ser usado Xamarin (para perceber melhor o porquê da escolha desta tecnologia ver secção 2.3.5). Para desenvolver em Xamarin vai ser utilizado o IDE Visual Studio [14] da Microsoft para programar em C#. A aplicação a desenvolver irá ser desenvolvida para as plataformas Android e Windows Mobile.

As aplicações desenvolvidas em Xamarin seguem o modelo MVVM (Model-View-ViewModel) [13] que é um padrão arquitetural de software muito usado pela Microsoft[7].

Para desenvolver as aplicações em Xamarin a Xpand IT usa várias *frameworks opensource*, nomeadamente MVVM Cross *framework open source* [13]. Uma das

[5] `https://www.android.com/` (acedido em Outubro 2016)

[6] Universal Windows Platform: `https://developer.microsoft.com/pt-pt/windows/getstarted` (acedido em Outubro 2016)

[7] `https://www.microsoft.com/pt-pt/` (acedido em Outubro 2016)

funcionalidades mais importantes desta *framework* é precisamente suportar *data binding* nas *Views* (basicamente permite criar a ligação entre dois objetos, sendo que a mudança de um desses objetos irá provocar a mudança do outro objeto).

1.3 Plano de trabalhos

Qualquer projeto de Engenharia de Software exige um planeamento dos trabalhos a ser realizados. Por esse motivo, segue-se a descrição detalhada dos planos de trabalhos do 1º e 2º semestre, com recurso ao diagrama de Gantt.

1.3.1 1º Semestre

Como já foi descrito na secção anterior (ver 1.2) existem algumas tecnologias a serem utilizadas para produzir o objetivo referido na secção 1.1. Tecnologias essas à qual na entrada para a Xpand IT não tinha os conhecimentos necessários. Como tal foram necessários alguns períodos de formação nas tecnologias e na metodologia usada na Xpand IT (assim como nas ferramentas usadas dentro da empresa para gestão de projetos - ver capítulo 3), seguida de fases de ambientação como podemos verificar no plano de trabalhos do 1º semestre (ver tabela 1.1 e imagem 1.1).

Tarefa	**Duração**	**Início**	**Fim**
Formação C#/Android	30 dias	10 Feb 2016	11 Mar 2016
Fase de ambientação (Android)	28 dias	11 Mar 2016	8 Apr 2016
Levantamento de requisitos	28 dias	25 Mar 2016	22 Apr 2016
Desenvolvimento Fase I (Android)	28 dias	22 Apr 2016	20 May 2016
Formação Windows	14 dias	20 May 2016	3 Jun 2016
Desenvolvimento Fase II (Windows)	14 dias	3 Jun 2016	17 Jun 2016
Relatório Intermédio	86 dias	6 Apr 2016	1 Jul 2016

Tabela 1.1: Plano de trabalhos 1º semestre

A primeira etapa consiste em formação de C# e Android. Por formação entende-se seguir um tutorial que a empresa disponibiliza para fazer uma aplicação básica em Xamarin (apenas a componente Android), permitindo assim assimilar melhor os conceitos. Para esta fase espera-se a duração de 30 dias. A segunda etapa consiste numa fase de ambientação à programação em Android, desenvolvendo algumas funcionalidades extra do tutorial que tinha sido feito na fase anterior. A

terceira fase passa por fazer o levantamento de todos os requisitos necessários para o desenvolvimento das funcionalidades a implementar na aplicação corporativa (objetivo desta tese). De seguida começa verdadeiramente o desenvolvimento das funcionalidades desenhadas na fase anterior, mas apenas na plataforma Android. No entanto, e porque o objetivo do Xamarin é desenvolver aplicações moveis para diversas plataformas, a intenção é desenvolver essas mesmas funcionalidades na plataforma Windows. Como tal existe novamente uma pequena fase de formação (cerca de 14 dias) mas desta vez para a plataforma Windows, seguida do desenvolvimento das funcionalidades para esta plataforma, ficando assim a aplicação em Android e Windows em paralelo. Por fim, e não menos importante, existe a escrita do relatório intermédio. Este irá começar aproximadamente a 6 de Abril com termo a 1 de Julho. Esta é a única fase que é executada em simultâneo com as outras tarefas para que o relatório esteja sempre a par da execução das tarefas.

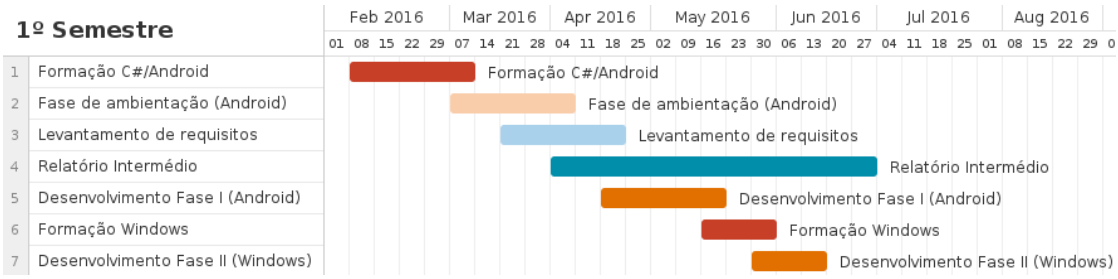

Figura 1.1: Gantt respetivo ao primeiro semestre

1.3.2 2º Semestre

Uma das componentes importantes no desenvolvimento de uma aplicação é a fase de testes que permite descobrir as possíveis falhas que possam existir. Como tal está prevista uma fase de formação em testes no início do segundo semestre, seguida dos verdadeiros testes à *release* da aplicação desenvolvida no primeiro semestre. Após esta fase de testes está prevista a correção dos eventuais bugs/-problemas encontrados e um novo *pack* de funcionalidades a desenvolver. No final de tudo está prevista uma nova fase de testes à nova *release* e por fim a escrita da tese final. Assim, e de uma forma geral está prevista a seguinte ordem de trabalhos a começar a 1 de Setembro de 2016:

- Formação em Testes;
- Realização de testes à *Release 1.0* (criação de *scripts*);

- Levantamento de novos requisitos para a *Release 2.0*;
- Desenvolvimento dos requisitos nas plataformas Android e Windows;
- Realização de testes à *Release 2.0* (criação de *scripts*);
- Escrita da tese final.

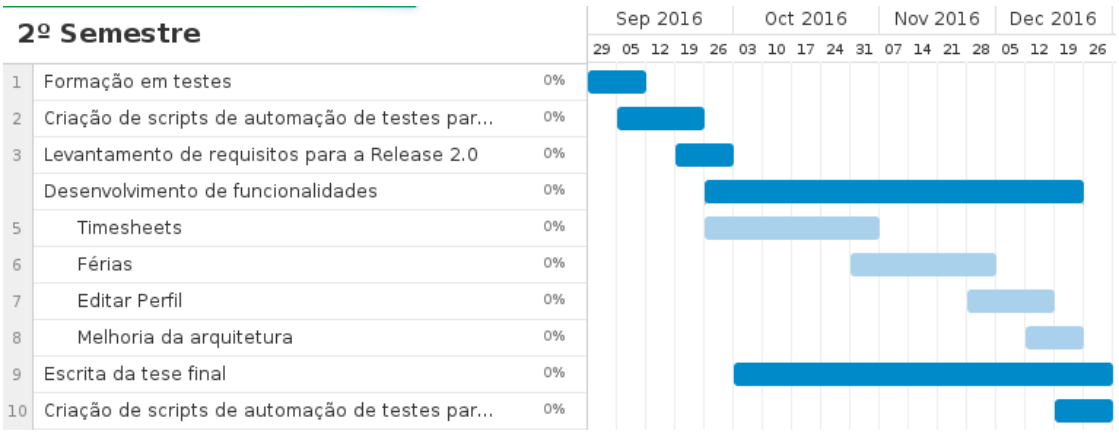

Figura 1.2: Gantt respetivo ao segundo semestre

Na figura 1.2 está retratado o Gantt respetivo ao segundo semestre. A seguir, na tabela 1.2 estão detalhadas as datas previstas para cada uma das tarefas.

Tarefa	Duração	Início	Fim
Formação em testes	8 dias	1 Sep 2016	9 Sep 2016
Criação de scripts para a R1.0	13 dias	10 Sep 2016	23 Sep 2016
Levantamento de requisitos para a R2.0	6 dias	24 Sep 2016	30 Sep 2016
Desenvolvimento de funcionalidades	80 dias	1 Oct 2016	20 Dec 2016
Criação de scripts para a R2.0	6 dias	21 Dec 2016	27 Dec 2016
Escrita da tese final	122 dias	1 Sep 2016	1 Jan 2017

Tabela 1.2: Plano de trabalhos 2º semestre

Capítulo 2

Estado da Arte

Durante os últimos anos o mercado *mobile* tem sofrido um enorme crescimento, assim como o desenvolvimento de aplicações *mobile*. Com estes acontecimentos surge um novo dilema: Devemos desenvolver aplicações de forma nativa, isto é, para cada uma das plataformas de forma separada, ou devemos optar por uma solução *cross-platform*? O que é mais vantajoso?

Aqui existem diversos fatores que têm de ser peremptoriamente analisados. Isto porque se tivermos uma aplicação publicada em todas as plataformas é sempre mais vantajoso do que ter apenas numa plataforma - estamos desta forma a abranger um maior número de potenciais clientes. No entanto, será que não existem desvantagens ao desenvolver aplicações para as diversas plataformas em simultâneo?

2.1 *Native versus Cross-platform*

Existem assim diversos fatores a ter em conta aquando da escolha do tipo de desenvolvimento [16]:

- **Tempo de desenvolvimento:** tempo estimado necessário para desenvolver a aplicação;
- ***Skills* do programador:** quantidade e qualidade de *skills* necessários para desenvolver aplicações;
- **Custo de desenvolvimento:** custo final do desenvolvimento da aplicação tendo em conta o tempo demorado e os recursos humanos utilizados.

Tudo depende do objetivo que existe para a aplicação. Considerando os vários requisitos, pode compensar optar por determinada alternativa em detrimento de outra.

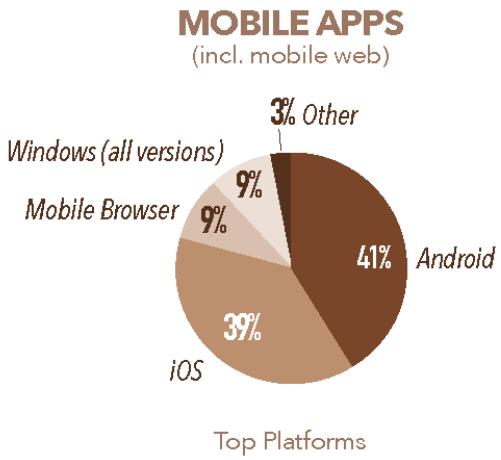

Figura 2.1: Top do desenvolvimento de apps para cada umas das plataformas
Source: `www.developereconomics.com` (acedido a Março 2016)

Na figura 2.1 temos a percentagem de aplicações desenvolvidas para cada uma das plataformas segundo um inquérito desenvolvido pela *Developer Economics* [6] de Outubro a Novembro de 2015 atingindo cerca de 21 000 programadores em mais de 150 países. Este estudo vem dizer que cerca de 41% das aplicações são desenvolvidas em Android e de seguida está o iOS com cerca de 39%. Windows segue-se com 9%, Mobile Browser com outros 9% e 3% para outros. Isto veio comprovar que o Android e o iOS em conjunto estão a ocupar uns esmagadores 80% da cota de mercado. Com esta análise obtém-se um *overview* das possíveis vantagens do panorama *cross-platform* visto que é muito mais vantajoso desenvolver uma aplicação nestas duas plataformas de uma única vez, do que desenvolver duas aplicações diferentes para estas duas plataformas distintas.

De uma forma geral na tabela 2.1 apresentam-se as principais vantagens e desvantagens da abordagem de desenvolvimento nativa comparada com a alternativa *cross-platform*. Como é conhecimento de todos a principal vantagem das alternativas *cross-platform* é o facto de reduzir o custo de desenvolvimento e conseguir obter um balanço qualidade/custo bastante razoável dado que temos uma compatibilidade multi-plataforma bastante elevada. Por outro lado, as Web Apps conseguem o menor custo de desenvolvimento, no entanto com um custo elevado na *"user experience"*, nas *features* e na *performance* que podem ser impactadas.

	Nativo	**Cross-platform frameworks**	**Mobile Web Apps**
Custo de desenvolvimento	+++	++	+
Compatibilidade multi-plataforma	- - -	++	+++
User experience	+++	++	-
Features e Performance	+++	++	+

Tabela 2.1: Comparativo Native versus Cross-platform development

2.1.1 *Native apps*

Aplicações desenvolvidas de forma nativa são aplicações que são construidas apenas para uma plataforma. Desta forma existem diversas vantagens subjacentes [18]. O desenvolvimento nativo já usa todas as *features* do dispositivo, isto é, inclui as API nativas disponibilizadas pela marca (Google, Apple ou Microsoft), logo está aqui garantida 100% de compatibilidade com os dispositivos sendo assim mais fácil de trabalhar e é garantida uma *performance* e qualidade superior da aplicação.

Entretanto, da perspetiva do utilizador as aplicações nativas não diferem muito das aplicações *cross-platform* em termos de qualidade. Porém é possível que as aplicações nativas sejam mais *user-friendly* visto que um *designer* quando está a desenhar a aplicação para uma plataforma específica já sabe quais os componentes existentes nesta, e quais os objetos mais adequados para cada tipo de operação [19]. Por exemplo, quando estamos a desenvolver uma aplicação para Android sabemos que podemos usar um sistema tabs superior e podemos fazer *"swipe"* para viajar entre as diferentes *activities*. Em iOS já não é possível fazer o mesmo exato *layout*[1] com as mesmas funcionalidades, tendo de optar por outro tipo de solução mais adequada.

No entanto, como não podia deixar de ser, existem também várias desvantagens. O desenvolvimento de aplicações de forma nativa tende a ser mais dispendioso já que exige uma equipa de um ou mais *developers* dedicada apenas para uma plataforma.

O processo de obter aprovação das aplicações nas *stores*, isto é, os reajustes que têm de ser feitos na aplicação (exigidos pela Google, Microsoft ou Apple) para esta ser aprovada, tem tendência a ser superior para os *developers*.

[1]Sem usar alguns truques para obter semelhanças.

Outra desvantagem é, no caso de produzir a mesma aplicação para as três diferentes plataformas de forma nativa, pode acontecer que as versões publicadas nas *stores*, sejam diferentes causando uma enorme inconsistência entre as plataformas e uma maior dificuldade por parte dos *developers* para manter o suporte às mesmas.

Falando em suporte, encontramos aqui uma das maiores desvantagens da produção de software nativo: **manutenção** - especialmente se a intenção for manter o suporte em mais que uma plataforma, já que o custo é extremamente elevado.

Posto isto, existem ainda algumas situações onde é aconselhado o desenvolvimento de aplicações de forma nativa [16]: por exemplo quando a *performance* é o requisito número um, ou quando é necessário processar uma enorme quantidade de informação na parte do cliente. Para cada tipo de solução é sempre necessário avaliar o método de desenvolvimento adequado aos requisitos impostos.

2.1.2 *Cross-platform apps*

Cross-platform é um conceito que merece esclarecimento. *Cross-platform* não significa que ao compilarmos uma aplicação para uma plataforma podemos usar o seu "executável"para instalar a aplicação noutra plataforma. Por exemplo, ao compilarmos uma aplicação para Android produzimos um ficheiro com extensão *.apk. Podemos tentar correr esse ficheiro num iPhone ou Windows Mobile, mas não iremos obter sucesso. Quer isto dizer que a vantagem do desenvolvimento de aplicações *cross-platform* concentra-se na otimização do processo de desenvolvimento (minimizar o custo de desenvolvimento). Na verdade o que acontece é que de facto é compilado o código fonte para executar nas várias plataformas mobile, mas o resultado é diferente para as três plataformas (por exemplo, para iOS é produzido um ficheiro com extensão *.ipa)[19].

Mas afinal o que atrai no desenvolvimento *cross-platform?*

Sem qualquer sombra de dúvida que a grande vantagem do desenvolvimento de apps *cross-platform* é o reduzido custo de desenvolvimento. Este tipo de abordagem permite desenvolver uma aplicação nas diferentes plataformas sem ter de investir numa enorme equipa de *developers*. A segunda grande vantagem é relativamente à manutenção que se torna bastante facilitada já que existe um código base partilhado pelas três plataformas em simultâneo, permitindo assim uma manutenção mais eficiente e rápida. Já para não falar que ao construir uma aplicação nesta abordagem faz com que as plataformas estejam automaticamente sincro-

nizadas, deixando para trás problemas como ter plataformas com versões mais atualizadas que outras. Desta forma todas as plataformas têm a versão mais atualizada da aplicação em causa.

Como desvantagem temos o facto de cada plataforma ter a sua particularidade. Isto significa que existem funcionalidades e/ou componentes que estão presentes em algumas plataformas e não existem noutras, levantando problemas quando pretendemos implementar um requisito ao longo das plataformas. No entanto, existe sempre forma de contornar a situação, tendo como custo o facto de nem todas as plataformas ficarem exatamente iguais a nível de UI e/ou de uma ou outra funcionalidade.

Outra desvantagem prende-se com problemas de integração a nível de armazenamento de dados. Diferentes plataformas têm diferentes abordagens de armazenamento podendo complicar a gestão a este nível. Porém existem *third-party services* que permitem aniquilar facilmente este problema, como por exemplo o Azure[2].

[2] *Cloud solution*, URL: `https://azure.microsoft.com/pt-pt/` (acedido a Abril de 2016)

2.2 Análise de abordagens

Nesta secção vou falar das diversas abordagens de desenvolvimento multi-plataforma. Na figura 2.2 está um resumo das três principais abordagens de desenvolvimento que vão ser exploradas nas próximas sub-secções.

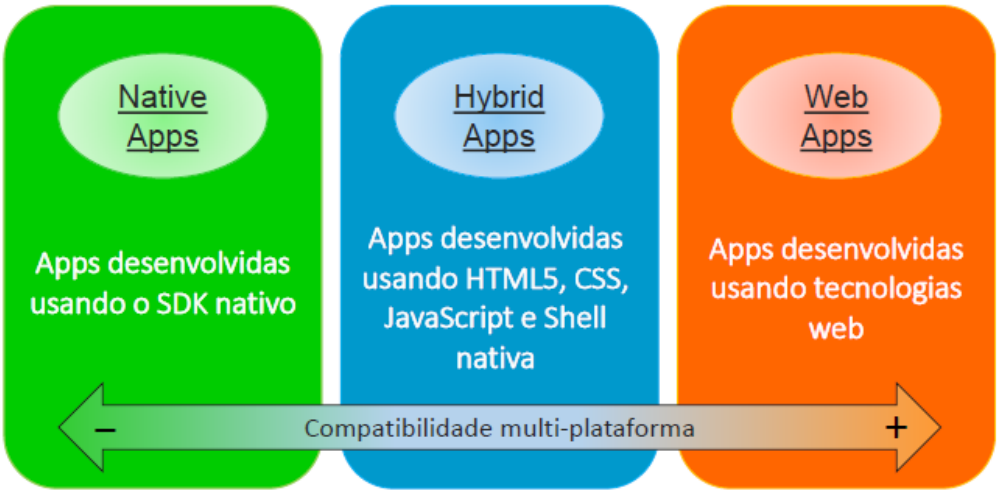

Figura 2.2: Abordagens de desenvolvimento de aplicações móveis

2.2.1 Web apps

O desenvolvimento de *"web apps"* é considerado um tipo de desenvolvimento multiplataforma dado que é possível correr uma página web através de qualquer uma das plataformas mobile. Segue o principio *Write Once Run Anywhere* (WORA) - Escrito uma vez, Executado em qualquer lado - reduzindo o tempo necessário para implementar a solução e reduzindo os custos envolvidos. Assim sendo, o número de potenciais utilizadores é bastante elevada face a outras alternativas *cross-platform*. O processo de desenvolvimento é muito equivalente ao desenvolvimento de uma página web o que traz só por si bastantes vantagens já que grande parte dos programadores tem noções de tecnologias web como HTML, CSS e/ou JavaScript [8].

Um pormenor a apontar desta alternativa é que não é possível fazer *deploy* das web apps nas diferentes *platform stores*. Assim, para aceder à aplicação é usado um *web browser*, não sendo necessário qualquer tipo de instalação nem ocupando espaço de armazenamento nos dispositivos móveis.

Um ponto que não favorece esta alternativa é a nível de UI. Desenvolver uma aplicação para correr em todas as plataformas torna a tarefas dos designers mais complicada. Isto porque, como já foi dito anteriormente, cada plataforma tem os seus padrões de UI. O que poderá ser bom para uma plataforma, pode ser negativo para outra.

Outro ponto negativo está relacionado com algumas restrições desta alternativa. Existem *frameworks web* que permitem a utilização de determinadas funcionalidades dos *smartphones* como a câmara, o GPS, ou até abrir outras aplicações. No entanto, o acesso às APIs da plataforma é limitado, e pode potencialmente ficar comprometido o acesso ao *hardware* e/ou a outras funcionalidades restritas da plataforma.

Porém, no meio de todas estas desvantagens existe ainda uma que se destaca pela negativa: *performance*. Sendo que a lógica de negócio desta alternativa se baseia em JavaScript, a performance está comprometida quando comparado com desenvolvimento nativo, já que a aplicação está limitada a um *browser* que por si só não oferece a mesma *performance* que uma aplicação. Ao optar por uma web app temos de estar cientes que a *user experience* vai ser limitada, até porque a aplicação apenas funciona *online*. Estando esta dependente da Internet, também está dependente da velocidade da rede a que estamos ligados. Quanto mais degradada for a velocidade da Internet, mais degradada irá ser a *performance* da app.

2.2.2 Hybrid apps

Aplicações híbridas são aplicações banais como tantas outras do ponto de vista que podemos procurar nas *app stores* e instala-las no nosso *smartphone* como qualquer outra app. A diferença prende-se com o facto da aplicação não passar de uma *web view* alojada numa aplicação nativa. Tal como as *web apps* (ver secção 2.2.1), as aplicações híbridas são desenvolvidas a partir das tecnologias HTML, CSS e JavaScript e usam uma API que providencia acesso a funcionalidades como o acelerometro, câmara, contactos, entre outros (funcionalidades estas que geralmente um *browser* normal não tem acesso) [3].

No processo de desenvolvimento de uma *hybrid app*, é produzido o executável para ser *deployed* para as plataformas e distribuído pelas respetivas *stores*. No entanto, tal como foi falado na secção anterior, a *performance* de uma aplicação híbrida encontra-se ainda bastante distanciada de uma aplicação nativa já que corre sobre JavaScript. Porém, consegue tipicamente ser superior às *web apps*

visto que parte do seu conteúdo está alojado na aplicação (no armazenamento do *smartphone*) ao invés de ser carregado através da Internet (com todos os inconvenientes que esta acarreta).

2.2.3 Interpreted apps

Interpreted apps são aplicações que não diferem muito das alternativas anteriores, no entanto, estão um passo à frente do ponto de vista que conseguem transmitir a sensação de estar a lidar com uma aplicação nativa.

Digamos que, por exemplo, a aplicação é escrita em JavaScript. O que acontece é que o JavaScript é executado no interpretador e este internamente executa chamadas para as APIs nativas (código nativo). O problema está que apenas se pode mapear código para um subconjunto que está disponível nas plataformas alvo. Isto pode ser visto como uma camada de abstração que permite ao programador escrever o código e fazer *deploy* nas plataformas suportadas. O acesso a APIs nativas permite o uso de UI nativa garantindo um visual mais apreciado pelos utilizadores, usando na mesma tecnologias web [8].

Quanto à *performance*, esta é tipicamente melhor que as aplicações híbridas mas obviamente não consegue atingir a velocidade de uma aplicação nativa. Apesar da interface ser nativa, a lógica de negócio está no interpretador que só por si irá degradar a *performance* [20].

2.2.4 Cross-Compiled

Um *cross-compiler* é um compilador que tem a capacidade de produzir código executável para uma plataforma que não a do compilador (onde este se encontra a correr). Temos como exemplo a Xamarin: temos o compilador a correr num ambiente Windows e a partir daqui irá ser produzido código executável para Android (plataforma esta diferente da do compilador).

Assim, *cross-compiled apps* são aplicações que foram produzidas a partir de uma plataforma diferente da plataforma alvo. Sempre que é necessário produzir uma aplicação para uma plataforma diferente é portanto necessário usar um *cross compiler* [4].

2.3 Análise de alternativas *cross-platform*

Apesar da tecnologia a ser usada no âmbito deste estágio ter sido estipulada para ser Xamarin, decidi analisar o que mais existe no mercado para validar se esta é realmente a melhor opção para o desenvolvimento deste projeto face aos requisitos que são impostos. Será que Xamarin é realmente a melhor opção? Será que não existe outra opção que permita desenvolver este projeto de uma forma mais rápida e eficiente?

Posto isto e dadas as abordagens analisadas na secção anterior (2.2), irei agora comparar cada uma das principais soluções *cross-platform* atualmente existentes no mercado.

2.3.1 Cordova e PhoneGap

PhoneGap é uma solução *cross-platform* com uma abordagem *"hybrid"* (ver 2.2.2) desenvolvendo aplicações *cross-platform* para as plataformas iOS, Android, Windows, Blackberry, Ubuntu e Firefox OS. Pertence à Adobe[3] e corre no topo do Apache Cordova[4] (apesar de cada plataforma ter o seu *PhoneGap engine*) que é uma *framework* de desenvolvimento *mobile* para obter acesso a determinadas funcionalidades dos *smartphones*, como o acelerometro, câmara, contactos, geolocalização, notificações, armazenamento, entre outros [8]. No entanto, mesmo assim o acesso ao *hardware* está limitado e o PhoneGap não permite o uso de funcionalidades como o sensor de luminosidade, sensor de proximidade e o giroscópio. Tem como vantagem criar aplicações usando HTML5, CSS3 e JavaScript sem depender de APIs específicas [15].

Outra particularidade do PhoneGap é que depois do desenvolvimento de uma aplicação estar completo, é necessário pagar uma taxa à PhoneGap para ser gerada a *final release* e consequentemente publicar a aplicação nas *application stores*. Por vezes algumas *stores*, nomeadamente a da Apple, rejeitam estas aplicações por se parecerem demasiado com uma página web, serem lentas e *"unresponsive"*, não suportarem um modo *offline* ou não escalarem de forma apropriada a resolução de imagem (problemas de UI). Assim é importante que o programador faça um bom trabalho, caso contrário irá acabar com uma aplicação lenta e sem aprovação das *stores*. Vale a pena referir que a PhoneGap/Cordova não têm neste momento suporte para *wearables* [11].

[3] `http://www.adobe.com/` (acedido a Abril de 2016)
[4] `https://cordova.apache.org/` (acedido a Abril de 2016)

2.3.2 Appcelerator Titanium

Appcelerator Titanium é uma *framework opensource* interessante que segue uma abordagem de modelo interpretado (ver 2.2.3) para desenvolver apps em iOS, Android, BlackBerry e Windows [2]. É baseada em JavaScript e usa HTML5, CSS3 e jQuery [12].

Esta *framework* promete aos programadores o desenvolvimento de aplicações *cross-platform* de uma forma fácil e rápida, resolvendo parte das dores de cabeças destes, escondendo todos os detalhes de implementação por de trás de uma API de JavaScript. De uma forma geral, o que a API irá fazer é compilar o código JavaScript para outra linguagem diferente dependendo da plataforma (Objective-C para iOS, Java para Android, etc.) para posteriormente fazer *deploy* para as múltiplas plataformas pretendidas, sem ter que mudar o código JavaScript. Para além disso é uma alternativa gratuita, inclusive para uso comercial (apenas alguns extras é que são pagos, como suporte e ferramentas de *testing*) [5]. As apps são desenvolvidas num ambiente Eclipse[5] (denominado Titanium Studio neste caso) e tem uma fácil integração com o Alloy[6].

Então quais são afinal as desvantagens desta abordagem? Aparentemente esta é uma ótima alternativa, basta escrever algum código JavaScript e temos uma aplicação nativa a funcionar. O problema é que quanto mais complexa a aplicação ficar, mais problemas esta irá ter. A aplicação começa a *crashar* de forma aleatória, sendo que a razão principal é *memory starvation*[7]. É suposto que o Titanium tome conta da gestão de memória por nós, porém essa gestão nem sempre funciona corretamente (ou da melhor forma). Quando a aplicação é simples e com poucas animações/imagens, não iremos sentir problemas com essa gestão de memória. O problema acontece quando o nível de complexidade da app começa a aumentar e estes pequenos problemas de gestão começam-se a tornar evidentes sob a forma de súbitos *crashes* inevitáveis. Pior de tudo é que não é possível fazermos nós a gestão de memória já que a Titanium não o permite (excetuando alguns truques que podem ser usados para reduzir o consumo de memória mas que maior parte das vezes não têm muito sucesso), portanto não existe outra solução senão deixar que a memória seja gerida automaticamente, podendo nos deixar assim com uma

[5]Eclipse é um IDE para desenvolvimento usualmente em Java. Saber mais em `https://eclipse.org/` (acedido a Abril de 2016).

[6]Alloy é uma *framework* MVC e *Cloud Service* para um desenvolvimento mobile backend pronto a usar. Saber mais em `https://wiki.appcelerator.org/display/guides2/Alloy+Concepts` (acedido a Abril de 2016)

[7]*Memory starvation* significa que existe um consumo excessivo de memória, podendo o sistema entrar em colapso.

aplicação frágil e pronta a *crashar* a qualquer momento [5].

Assim sendo, se não pretendemos desenvolver uma aplicação muito complexa, esta alternativa pode ser uma ótima solução já que simplifica muito o trabalho dos *developers*. Se, pelo contrário, pretendemos construir algo mais complexo é aconselhado outro tipo de alternativa.

2.3.3 Sencha Touch

Sencha Touch pertence à empresa Sencha, e foi criada com a intenção de facilitar a vida aos *developers* que já estavam habituados a usar o ExtJS (*framework* da Sencha para criar aplicações com uma UI complexa), para que pudessem também desenvolver aplicações móveis [10]. Sencha Touch é uma *framework* MVC *open-source* de JavaScript desenvolvida para o paradigma *web mobile*. É baseada nas *web standards* tais como HTML5, CSS3 e JavaScript e permite criar *web apps* para as plataformas Android, iOS, Windows, BlackBerry, entre outras, produzindo uma sensação de aplicação nativa apesar de não ser mais do que uma *web-view*. Em termos de APIs, o Sencha Touch está integrado com o PhoneGap/Cordova (ver 2.3.1) e como tal tem acesso às funcionalidades que este permite (como o acelerometro e a câmara) [17].

Assim, em termos de desvantagens, esta segue de uma forma uniforme os mesmos padrões de uma típica *web app*, isto é, *performance* degradada seja pelo facto do código JavaScript que não está otimizado ou pela simples razão de ser uma *web app* (ainda que com alguns traços de uma *hybrid app* derivado da utilização das APIs do PhoneGap/Cordova).

2.3.4 Xamarin

A Xamarin é uma alternativa que segue uma abordagem *cross-compiled* (ver 2.2.4) para criar aplicações nativas para as plataformas Android, iOS, e Windows. O objetivo principal passa por desenvolver apps verdadeiramente nativas mas em que possa existir um "core"de código em comum entre as três plataformas (ver Fig. 2.3). A ideia da Xamarin é dividir a implementação em duas camadas: a camada de negócio onde está o código partilhado entre as plataformas suportadas e a camada de *user interface*. Desta forma é reutilizado cerca de até 75%[8] do

[8]Informação disponibilizada pela própria Xamarin no *website* deles, URL: `https://www.xamarin.com/platform` (acedido a Junho de 2016)

código. É usado apenas C# para programar ao longo das três plataformas, o que é uma grande vantagem já que temos acesso a toda a *framework* .NET (LINQ, Tasks, etc.) que simplifica bastante o trabalho do programador.

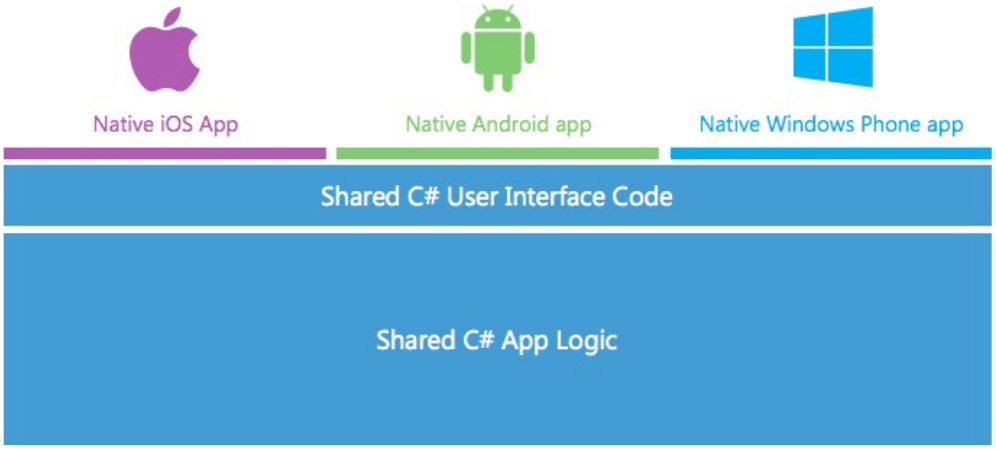

Figura 2.3: Lógica de negócio da Xamarin
Source: `www.xamarin.com` (acedido a Março de 2016)

As aplicações em Xamarin podem ser desenvolvidas tanto no OS X como nos computadores Windows (apesar de os utilizadores do OS X terem a restrição de não puderem desenvolver aplicações para a plataforma Windows) usando o Xamarin Studio (IDE desenvolvido pela Xamarin) que simplifica a publicação e a distribuição das aplicações.

Apesar da Xamarin disponibilizar o Xamarin Studio, também tem integração com o Microsoft Visual Studio através de um *plugin* que permite o desenvolvimento de aplicações iOS e Android neste IDE. A compilação de aplicações iOS têm a particularidade de terem de ser feitas numa máquina OS X (devido ao facto do compilador do iOS não ser *open source*). No entanto a Xamarin tornou este processo mais fácil através de um mecanismo que faz uma ligação remota a uma máquina OS X local apenas para fazer a compilação do código escrito num computador Windows.

Para que não exista nenhuma lacuna nas funcionalidades específicas de cada plataforma, a Xamarin criou um *"mapping"* completo das APIs de Android e iOS. Desta forma, é possível construir uma UI usando componentes nativos, fazendo com que a app se comporte como se tivesse sido construida de forma nativa. Através deste *mapping* de APIs, é possível aceder a qualquer funcionalidade seja a nível de *hardware* ou de *software* (integração com outras apps por exemplo) como

qualquer outra aplicação nativa. Vale a pena referir o Xamarin.Forms[9] como uma abordagem que está em curso mas que permitirá no futuro endereçar o tema da UI da mesma forma que hoje em dia endereçamos a lógica de negócio.

Uma das grandes mais valias da Xamarin face a qualquer outra alternativa *cross-platform* é que uma aplicação desenvolvida em Xamarin é muito semelhante a uma aplicação nativa a nível de *performance* e UI, e como tal é tão facilmente aceite como qualquer outra aplicação nativa. Um grande exemplo disso é que a nível da UI conseguimos respeitar as *guidelines* específicas de cada uma das plataformas. Olhando para o exemplo dos *"Segmented Controls"* no Android e no iOS conseguimos comprovar precisamente isso (ver imagem 2.4). São duas opções distintas mas com o mesmo propósito respeitando o *design* de cada plataforma. Esta hipótese não se encontra disponível numa *web-app* visto que a UI fica igual em qualquer uma das plataformas.

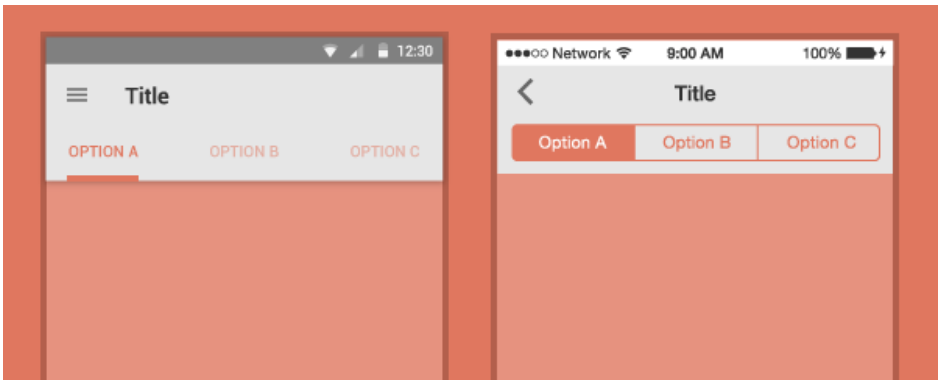

Figura 2.4: *"Segmented Controls"* em Android vs iOS respectivamente
Source: `https://webdesign.tutsplus.com/articles/a-tale-of-two-platforms-designing-for-both-android-and-ios--cms-23616`
(acedido a Março de 2016)

Passando agora aos contras desta alternativa: é certo que podemos partilhar código base, *business logic*, etc. - mas se nós precisarmos de customizar em pormenor a nossa UI, então temos de escrever em cada plataforma de forma independente já que não é possível fazer uma relação 1:1 entre atividades Android e iOS UIViewControllers, por exemplo. Outra desvantagem pretende-se com o Xamarin Studio no OS X: apesar de tudo, esta ferramenta ainda está bastante longe do Xcode ou do Android Studio. Tem vindo a melhorar cada vez mais ao longo dos tempos, e

[9]Xamarin.Forms: `https://www.xamarin.com/forms` (acedido em Janeiro de 2017)

funciona, claro, mas não tem as mesmas *"extra features"* comparativamente a outros IDEs nativos. No entanto, esta falha pode ser facilmente ultrapassada usando o Visual Studio [1].

2.3.5 Retrospetiva

No final de tudo, qual é a melhor alternativa *cross-platform*? A resposta é – depende. É difícil de indicar qual a melhor alternativa. Tudo depende do objetivo da aplicação que pretendemos desenvolver face aos requisitos. Por exemplo, caso o objetivo seja desenvolver uma aplicação apenas para Android em que o requisito número um é *performance*, então nesse caso nem vale a pena optar por uma alternativa *cross-platform*. Caso o objetivo seja desenvolver uma aplicação para um número elevado de plataformas em que a velocidade e estabilidade da app não seja o mais importante, então pode-se optar por uma alternativa do género do PhoneGap/Cordova. Caso o objetivo seja desenvolver uma app para Android, iOS e Windows em que tanto a *performance* como a UI são muito importantes, então a Xamarin acaba por ser a melhor solução. O importante a referir é que existem muitos fatores a serem ponderados antes da escolha de uma alternativa.

Funcionalidades	Cordova/Phonegap	Appcelerator Titanium	Sencha Touch	Xamarin
Suporte multiplataforma				
iOS	✓	✓	✓	✓
Android	✓	✓	✓	✓
UWP	✓	✓	✓	✓
Acesso API's nativas	x	x	x	✓
Linguagem de Desenvolvimento	HTML5, CSS3, JS	HTML5, CSS3, JS, jQuery	HTML5, CSS3, JS	C#
Abordagem	Hybrid	Interpretado	Web	Cross-compiled
Curva de aprendizagem	Elevada	Elevada	Elevada	Média
Look and Feel	simulado	simulado	simulado	nativo
Performance	deteriorada	deteriorada	deteriorada	elevada
Estabilidade	deteriorada	deteriorada	deteriorada	elevada
Offline Storage	✓*	✓	✓*	✓
Hardware and data access	Limitado	Limitado	Limitado	Completo
Comunidade	+++	+	+++	+++

*A funcionalidade *offline storage* é limitada.

Tabela 2.2: Comparativo de alternativas multiplataforma

Na tabela 2.2 está retratado um resumo das características de cada uma das

alternativas multiplataforma analisadas anteriormente. Em qualquer uma das alternativas é possível exportar a nossa aplicação para iOS, Android e UWP. No entanto, Xamarin é a única alternativa "não-web"e que tem acesso a 100% das API's nativas. Tem um custo de aprendizagem superior às outras alternativas mas compensa com uma UI nativa, e uma *performance* e estabilidade da aplicação muito superior às concorrentes. Para além disso, é também a única alternativa que consegue usufruir da utilização de todo o *hardware* dos dispositivos (giroscópio, GPS, etc.).

Posto isto, a *Developer Economics*[10] fez a recolha do top das alternativas *cross-platform* utilizadas em 2015 em que as três primeiras no top são a escolha número um de 70% dos inquiridos (ver Fig. 2.5). Assim no top 3 está o PhoneGap/Cordova em primeiro lugar – a ganhar vantagem face às restantes alternativas, o Xamarin em segundo e o Unity em terceiro lugar. As restantes alternativas, por ordem respetivamente, Qt, Adobe Air, Appcelerator, Corona, Marmelade, Codename One e por fim LiveCode. O facto do PhoneGap/Cordova estar em número um pode-se justificar pelo facto de haver um grande número de aplicações em que nos requisitos não se destaca nem a *performance*, nem a UI. No entanto logo a seguir encontra-se a Xamarin que acaba por ser a melhor alternativa para quem pretende fazer uma aplicação nas três plataformas sem sacrificar determinados requisitos.

[10] `http://www.developereconomics.com/` (acedido a Abril de 2016)

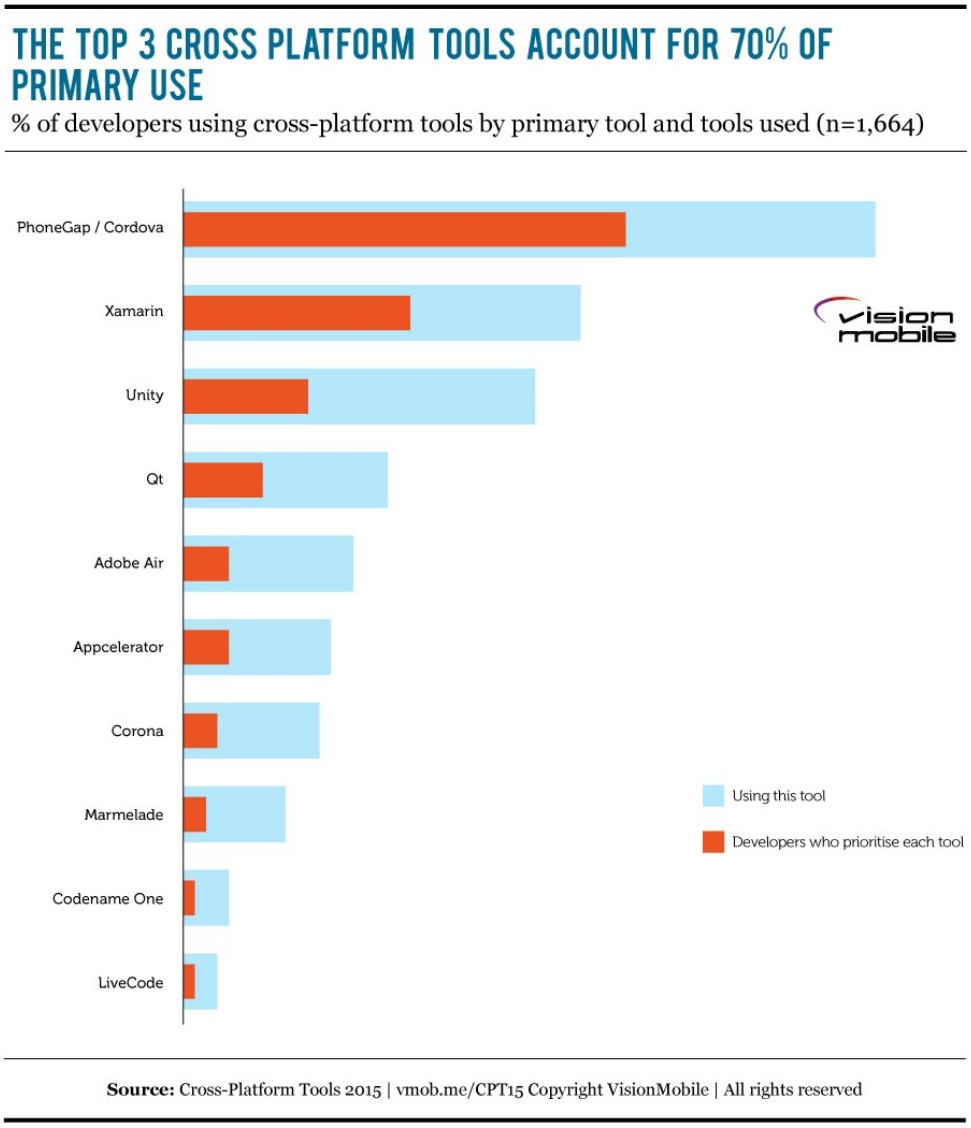

Figura 2.5: Top das alternativas cross-platform utilizadas em 2015

Por último, e para ressaltar a importância destas alternativas *cross-platform* – principalmente falando da Xamarin – dominar uma linguagem por vezes torna-se uma tarefa complicada, então imagine-se dominar três linguagens diferentes (neste caso Java, Objective-C e C#). Para quê programar duas ou três vezes o mesmo código quando o pudemos fazer apenas uma vez numa única linguagem de programação? Já para não falar que para além de dominar uma linguagem, é necessário nos mantermos atualizados à cerca das novas *frameworks*, versões, livrarias, etc. [1].

Deste modo, a Xamarin foi a selecionada para o desenvolvimento do projeto da presente tese. Esta escolha justifica-se pelo facto da aplicação ter como alvo os três ecossistemas dominantes em termos de ambientes móveis (Android, iOS e Windows), satisfazendo a necessidade de proporcionar uma UI coerente entre as versões desenvolvidas para cada plataforma (respeitando as *guidelines* de cada plataforma), sem deixar de assegurar níveis de usabilidade e desempenho adequados - razão pela qual não foi escolhida uma *web-app*.

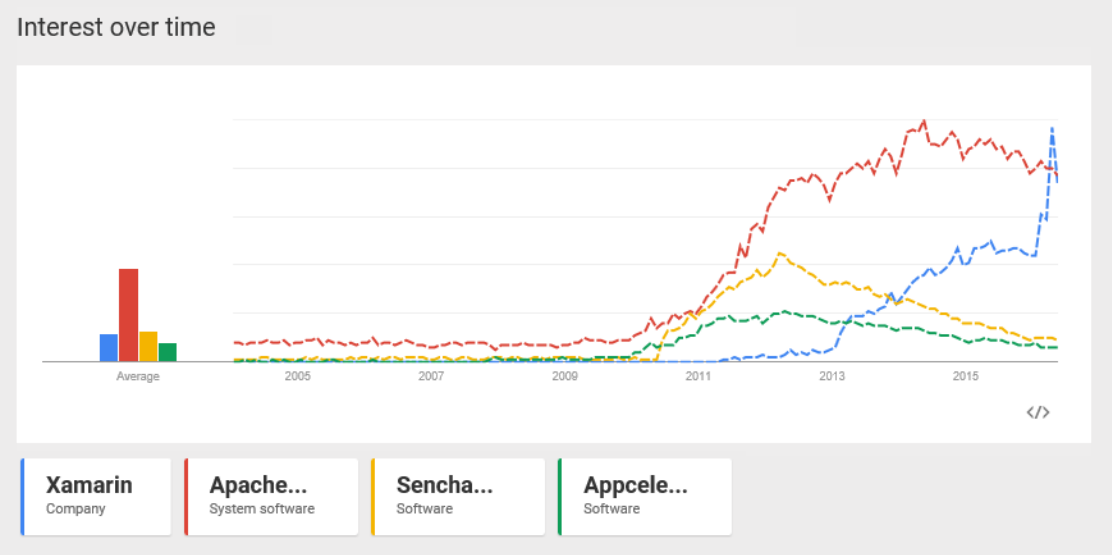

Figura 2.6: Interesse nas alternativas de desenvolvimento multi-plataforma segundo a Google Trends
Source: `www.google.pt/trends` (acedido a Abril de 2016)

Por curiosidade e em formato de conclusão, na figura 2.6 está ilustrado o interesse em algumas das alternativas multi-plataforma que foram analisadas anteriormente (Xamarin, Cordova/PhoneGap, Sencha Touch e Appcelerator Titanium) segundo as pesquisas que foram feitas na Google, usando a Google Trends[11]. Podemos retirar daqui primeiro de tudo, que o interesse nas abordagens multi-plataforma já existem há algum tempo, tendo-se no entanto acentuado em finais de 2010. Porém, a alternativa que mais se destaca é o PhoneGap que começou a crescer de forma acentuada em 2011 tendo normalizado a partir de 2015. Por outro lado, quem tem demonstrado um grande crescimento nos últimos anos (sensivelmente desde 2013) tem sido a Xamarin que em 2016 conseguiu ultrapassar a

[11]Google trends, url: `https://www.google.pt/trends` (acedido a Abril de 2016)

PhoneGap, o que leva a crer que realmente esta alternativa está a ganhar cada vez mais terreno nos dias de hoje. Vale a pena referir que este crescimento acentuado em 2016 também está relacionado com a aquisição pela Microsoft que acabou por legitimar a abordagem da Xamarin com uma grande marca por trás, e que por isso mesmo despertou a comunidade de *developers* .Net para esta alternativa. As restantes duas alternativas (Sencha Touch e Appcelerator Titanium) tiveram algum crescimento, no entanto não muito acentuado, tendo decrescido o interesse a partir de 2013. Por último, vale a pena referir que existe uma alternativa que está a começar a ganhar espaço para evoluir e talvez venha a ser uma alternativa interessante – React Native[12] – que promete o desenvolvimento de aplicações nativas em JavaScript e React.

2.4 Aplicações semelhantes no mercado

Para se justificar o desenvolvimento desta aplicação foi analisado o mercado em busca de soluções semelhantes que conseguissem satisfazer as necessidades sentidas pelos colaboradores da Xpand IT.

Existem assim diversas aplicações que como um todo conseguem complementar as funcionalidades da Xpack:

- **Lista de contactos:** a lista de contactos dos colaboradores da empresa pode ser comparada, por exemplo, à lista de contactos do Outlook[13] que permite aceder não só a todos os nossos contactos como saber se alguém faz anos num dado dia (com notificações) tal e qual como na Xpack; No entanto os contactos têm de ser adicionados manualmente, e consequentemente, pedir pessoa a pessoa já que não existe nenhuma fonte acessível com toda a lista dos contactos da empresa;
- **Confluence:** existe atualmente uma aplicação do Confluence que satisfaz as necessidades existentes na aplicação. No entanto existem dois contras: a informação não está tão acessível como na Xpack, e a app não se encontra disponível para UWP;
- **Jira:** existe também a aplicação do Jira disponibilizada pela Atlassian[14],

[12] `https://facebook.github.io/react-native/` (acedido em Janeiro 2017)

[13] Android: `https://play.google.com/store/apps/details?id=com.microsoft.office.outlook&hl=pt_PT` (acedido a Abril de 2016), iOS: `https://itunes.apple.com/us/app/microsoft-outlook-email-calendar/id951937596?mt=8`(acedido a Abril de 2016)

[14] `https://www.atlassian.com/` (acedido a Abril de 2016)

no entanto nós como colaboradores da Xpand IT não conseguimos aceder através da aplicação deles visto que o Jira não se encontra alojado na Cloud da Atlassian mas nos nossos servidores;

- **Mapas:** podemos eventualmente comparar o redirecionamento da morada dos escritórios da Xpand IT com o Google Maps por exemplo; no entanto não creio que seja uma comparação correta visto que a Google Maps não disponibiliza a morada completa de cada um dos escritórios em poucos *clicks*, nem tão pouco disponibiliza informação como os números de telefone e fax da empresa;

Todas as restantes funcionalidades não são encontradas em apps *mobile* no mercado existente atualmente. Estamos a falar das seguintes funcionalidades:

- **Informação Fiscal:** não existem aplicações cujo foco seja disponibilizar a informação fiscal das empresas;
- **Férias:** as férias são atualmente feitas através de um ficheiro Excel, no entanto estão a ser exportadas para o serviço Timelive[15], serviço este que não disponibiliza qualquer aplicação *mobile*; podíamos eventualmente comparar com outras aplicações de férias atualmente existentes no mercado, mas não iria permitir a comunicação com este serviço já que são serviços isolados e não integrados com o sistema da Xpand IT;
- **Timesheets:** o serviço de timesheets é feito também ele através do Timelive, razão pela qual não existe alternativa senão o desenvolvimento de uma nova app;

De uma forma geral e em formato de conclusão, não existe nenhuma aplicação *mobile* no mercado que permita o agregamento de todos estes serviços e informação disponível numa única app. Estão assim reunidas todas as condições para o desenvolvimento de uma aplicação que irá facilitar o dia-a-dia de qualquer colaborador da Xpand IT.

[15] `http://www.livetecs.com/` (acedido a Abril de 2016)

Capítulo 3

Metodologia de desenvolvimento

Em engenharia de *software* um dos processos essenciais na gestão de um projeto é a existência de uma metodologia de desenvolvimento que explicitamente diga como é que todo o processo de desenvolvimento do *software* se vai desenrolar. Estas metodologias existem para simplificar o processo de desenvolvimento (que é complexo por natureza) modelando como o projeto é planeado, controlado e monitorizado desde o início ao fim, tentando assim evitar potenciais fontes de risco.

Existem assim diversas opções para metodologias de desenvolvimento, tais como *ad hoc* (o menos aconselhado – do-test, do-test, ...), *waterfall* (o clássico com as fases em cascata já muito conhecidas – análise de requisitos, projeto, implementação, testes (validação), integração, e manutenção de software), incremental (estratégia de planeamento estagiado em que várias partes do sistema são desenvolvidas em paralelo, e integradas quando completas), espiral (mistura o modelo "*Prototype*"com o controle e sistematização do modelo *waterfall*), entre outros. No entanto, estas metodologias clássicas são demasiado exigentes no que toca a requisitos e documentação e não respondiam de forma eficiente ao dinamismo e flexibilidade que a realidade do mercado pedia, acabando muitas vezes por não corresponder às necessidades reais do cliente.

Eis que surge então uma outra metodologia – *Agile* (ou Ágil) – cujos princípios se regem pela a flexibilidade e facilidade de mudança. Esta é a metodologia usada nos projetos desenvolvidos dentro da Xpand IT (pelas razões que iremos ver a seguir) e foi também a metodologia usada no desenvolvimento do projeto da presente tese. Esta metodologia aproxima a equipa de desenvolvimento, com os clientes finais (seja ele interno ou externo). A homologação dos projetos é feita em etapas,

resultando assim em tempos de entrega mais curtos e capacidade de promover alterações rapidamente. Na imagem da Fig. 3.1, podemos ver a balança dos valores *agile*, existem valores que ganham notoriamente mais peso:

- **Foco nas pessoas e interação em vez dos processos e ferramentas**, já que a comunicação e colaboração são fundamentais para o sucesso de um processo de âmbito dinâmico, não desprezando os processos e as ferramentas (elementos estes importantes, no entanto, insuficientes para garantir o sucesso do projeto);
- **Foco no software funcional em vez da documentação exaustiva**, permitindo aos clientes experimentar um pacote totalmente funcional e continuamente analisar se corresponde às suas reais necessidades, tendo na mesma documentação que é imprescindível, no entanto concisa e clara, tendo como propósito suportar a execução do projeto e não para proteção do âmbito;
- **Foco na colaboração do cliente em vez de negociação de contractos**, promovendo o envolvimento dos clientes nas fases de validação e definição de âmbito dos próximos *sprints* – o regular e frequente *feedback* do cliente durante o ciclo de desenvolvimento elimina o risco de surpresas na entrega, pois o alinhamento de expectativas foi trabalhado de forma contínua;
- **Foco na resposta à mudança em vez de seguir um plano rígido**, sendo que a mudança é uma realidade no desenvolvimento de *software*, é necessário que a metodologia reflita este requisito, apesar de haver na mesma um plano – mas flexível – com espaço para mudanças (caso contrário o mesmo tornar-se-á desatualizado e pouco útil).

Na metodologia usada na gestão dos projetos na Xpand IT (e no presente projeto), são também usados alguns conceitos do Scrum, nomeadamente conceitos de desenvolvimento iterativo e incremental, visto que rever as fases de desenvolvimento aumenta consideravelmente a eficiência do projeto. Assim, todo o desenvolvimento é dividido em iterações e cada iteração dividida nas tradicionais fases de desenvolvimento (que irão ser mais aprofundadas à frente - ver secção 3.1). Como podemos ver na figura 3.2, cada iteração do processo *agile* é constituído por uma fase de “Planeamento”, “Implementação”, “Testes”, “Avaliação”e de seguida voltamos novamente ao passo inicial.

Este método permite assim reduzir o risco, antecipar a descoberta, realização de testes antecipados (antecipada mitigação do risco) e a continua avaliação e

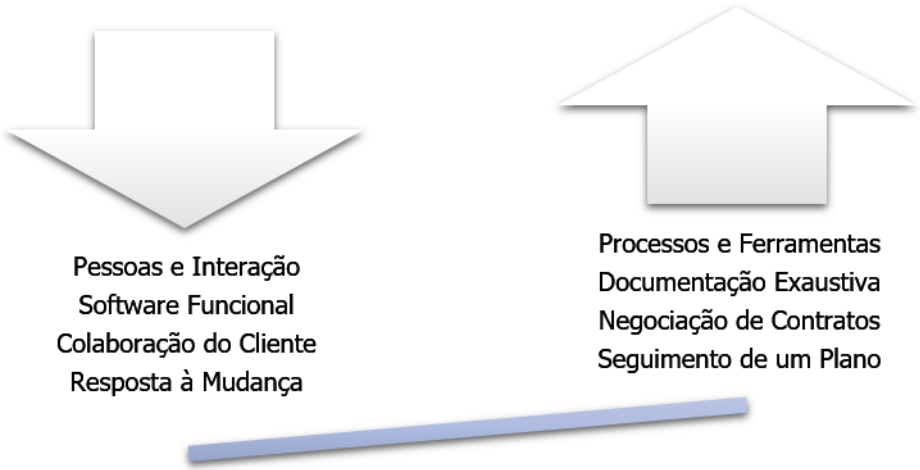

Figura 3.1: Balança dos valores *Agile*

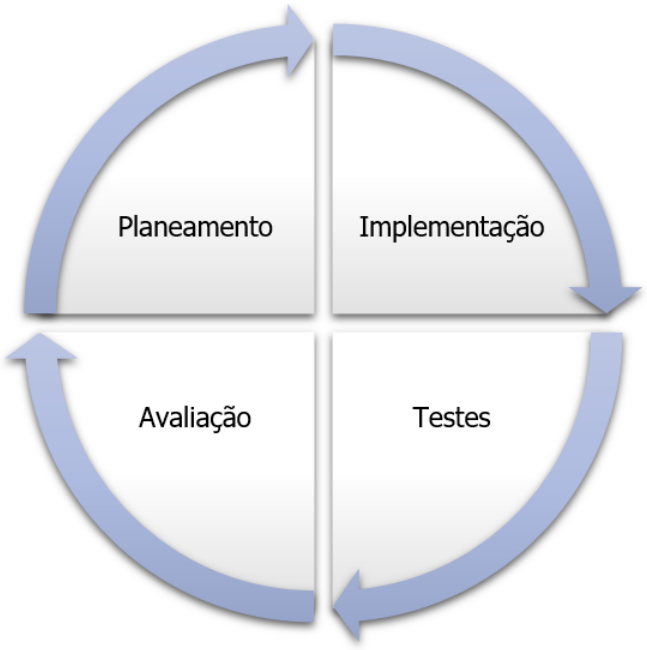

Figura 3.2: Desenvolvimento Iterativo e Incremental

feedback do cliente, resultando num produto com maior sucesso, e sobre tudo, mais adequado às necessidades do cliente. Para além disso, do ponto de vista dos clientes, estes podem deixar algumas decisões para mais tarde (para as quais ainda não há informação ou necessidade de resolução imediata). As decisões podem ser proteladas para as iterações futuras, quando houver melhor informação ou estrutura disponível, otimizando assim a eficiência das escolhas do cliente.

3.1 O processo

A metodologia adotada permite alguma flexibilidade a nível dos requisitos, ou seja, estes podem ser construidos ao longo do desenvolvimento do projeto não obrigando a definir em grande detalhe todos os requisitos logo no início.

Uma componente importante no desenvolvimento *Agile* é o foco nas funcionalidades que representam maior valor para o negócio . O custo da mudança é reduzido, o que representa uma grande oportunidade para o cliente reexaminar os fatores do negócio no início de cada iteração para selecionar as funcionalidades a incluir na corrente iteração. As equipas de desenvolvimento da Xpand IT alertam o cliente para o risco, no entanto, o cliente é quem decide o que a equipa deve fazer na respetiva iteração. Na figura 3.3 está retratado o processo desenvolvido nesta metodologia. Tudo começa com um *sprint* que corresponde a uma iteração do projeto (com duração entre 2 a 4 semanas) onde é implementado um conjunto de funcionalidades. Durante cada *sprint*, a equipa produz um incremento ao produto final. Nas próximas secções irá ser explicado cada etapa de um *sprint*.

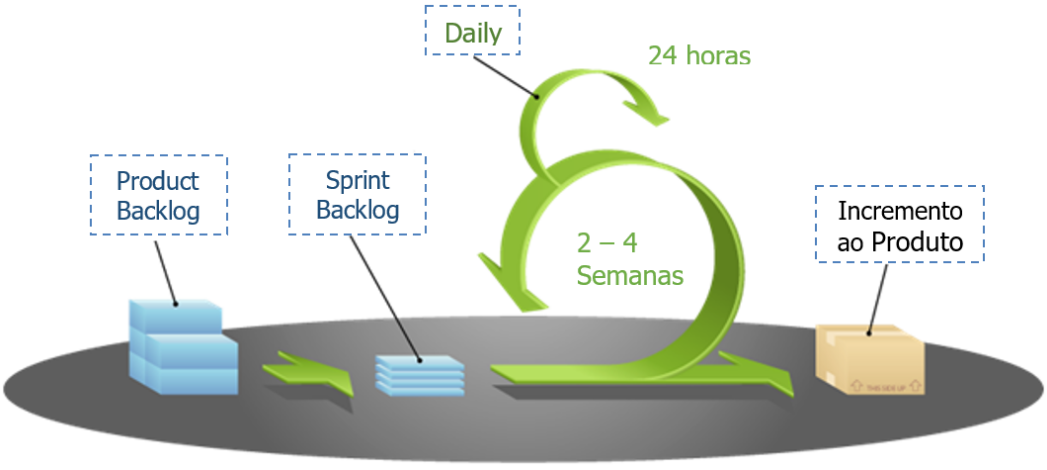

Figura 3.3: Processo Agile

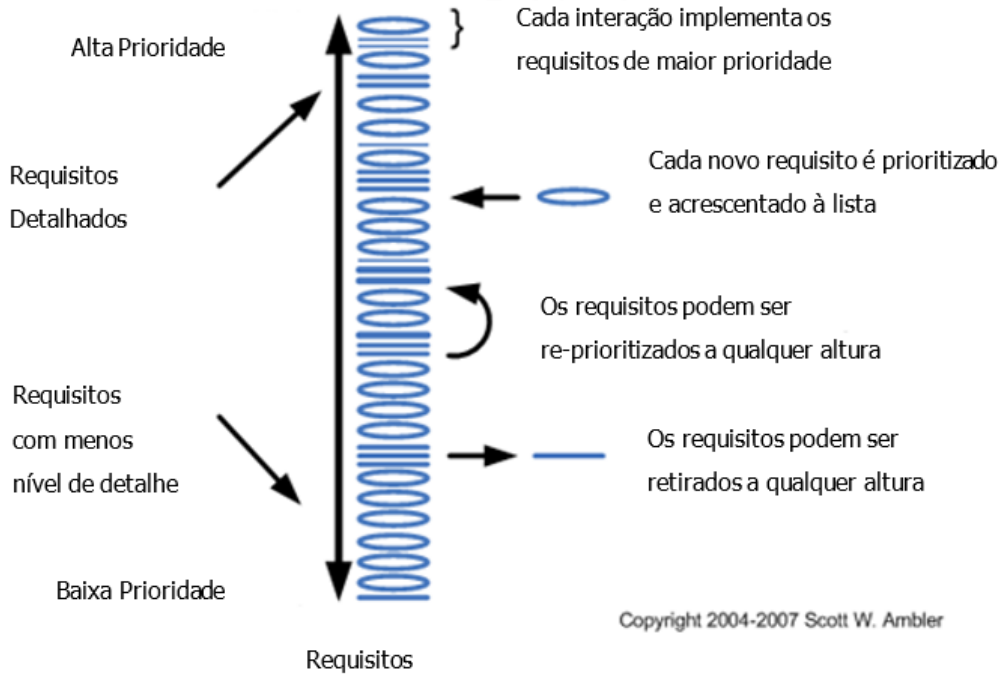

Figura 3.4: Product Backlog

3.1.1 *Product backlog*

O Product Backlog corresponde ao primeiro passo da iteração e corresponde a uma lista de funcionalidades (alterações desejadas para o produto) ordenadas por prioridade. Com esta abordagem as equipas de desenvolvimento têm uma pilha de tarefas ordenadas por prioridade e estimadas, que necessitam de ser executadas. Os *Stakeholders* são responsáveis por priorizar os requisitos; a equipa é responsável por estima-los e assim o *Product Backlog* está ordenado pelo valor.

Na figura 3.4 está representado o *Product Backlog*, onde no topo da pilha estão as funcionalidade descritas com um maior nível de detalhe em contraste com as funcionalidades que estão na base da pilha que não necessitam de uma descrição exaustiva. Isto porque só as funcionalidades a implementar na atual iteração é que necessitam de detalhe. No caso dos requisitos das futuras iterações só é necessário detalhe para determinar a sua estimativa.

3.1.2 *Sprint backlog*

O *Sprint Backlog* corresponde à lista dos requisitos que a equipa se compromete a desenvolver até ao final do atual *sprint* com base nas prioridades definidas em cada um dos elementos na pilha de requisitos (tendo em conta que são selecionados aqueles que tiverem uma prioridade mais elevada) e da perceção da equipa quanto ao tempo que vai demorar a completar cada uma das diversas funcionalidades.

3.1.3 Reunião diária (*Daily Meeting)*

Durante o decorrer de um *sprint*, a equipa reúne-se todos os dias (cerca de 15 minutos, de preferência à mesma hora) para que todos os membros desta fiquem a par do decorrer dos desenvolvimentos que aconteceram no dia anterior, e para que sejam discutidas soluções para possíveis problemas que tenham ocorrido ou possam ocorrer. É feito também um planeamento do que irá ser desenvolvido no dia corrente. Essencialmente cada elemento da equipa de desenvolvimento irá responder às seguintes questões:

- O que foi feito no dia anterior?
- O que está planeado fazer hoje?
- Existe algum impedimento para que as tarefas que estão planeadas não se cumpram?

3.1.4 Conclusão e retrospetiva do *sprint*

No final de cada *sprint* é realizado um *Sprint Review Meeting* que tipicamente inclui o cliente, a equipa e o gestor de projeto. Durante a reunião a equipa mostra o que concluiu durante o *sprint* e faz uma demonstração das novas funcionalidades.

Para além disso, a equipa reúne-se para refletir sobre o que funcionou, o que não funcionou e como o processo pode ser melhorado. Este encontro chama-se *Sprint Retrospective Meeting*. É um processo colaborativo entre os elementos da equipa e o gestor do projeto que deve garantir que a equipa está continuamente a melhorar a sua forma de trabalhar.

Depois de todos estes passos, e caso o produto ainda não se encontre concluído, um novo *sprint* tem início com base no *Product Backlog* existente e corrigindo os erros que possam ter acontecido no *sprint* anterior.

3.1.5 JIRA

Como apontamento, dentro da Xpand IT e neste projeto, foi usada uma ferramenta essencial no desenvolvimento de *software* (ferramenta esta obrigatória no desenvolvimento de qualquer projeto de *software* dentro da empresa). Essa ferramenta designa-se JIRA[1] e é uma ferramenta que permite a monitorização das tarefas e o acompanhamento do projeto, providenciando um local onde se encontram todas as atividades e alterações feitas no projeto. Todos os passos desta metodologia estão visíveis no JIRA, seja o *Product Backlog*, o *Sprint Backlog* ou até mesmo as *Daily Meetings* que têm de estar preenchidas nesta ferramenta. Em cada uma das tarefas a realizar está descrita a funcionalidade a ser desenvolvida acompanhada de um *mockup* do ecrã correspondente.

3.2 Metodologia no âmbito do projeto

Para o desenvolvimento do presente projeto foram usadas as regras desenhadas na metodologia descrita deste capítulo. A aplicação desenvolvida é para uso interno, isto é, destina-se aos colaboradores internos da empresa. Como tal o cliente são todos os colaboradores da Xpand IT, apesar de ser o *project manager* a representa-los para efeitos de funcionalidades a colocar no *Product Backlog*.

Na figura 3.5 está um exemplo de um *"screenshot"* feito à ferramenta no âmbito do desenvolvimento do projeto da presente tese. Nesta figura está retratada uma *issue* que estava no *Sprint Backlog* e que pertence agora ao *sprint 8*. Existe um *"Epic Link"* onde basicamente tem a *user storie* da presente *issue*. Todas as *issues* têm uma prioridade como já foi dito na secção anterior, que neste caso era *"critical"* (de elevada prioridade). Existe também um controlo de tempo, com uma estimativa do tempo que esta *issue* irá demorar a ser desenvolvida. Após estar concluída é registado o tempo que efetivamente demorou a ser desenvolvida para que futuramente exista um melhoramento das estimativas e mais facilmente avaliar a dimensão de um projeto. Para além disso, existe uma parte da página que tem o *url* para a *"Daily Meeting"* onde esta *issue* foi referida.

[1] JIRA Software, url: `https://www.atlassian.com/software/jira` (acedido a Maio de 2016)

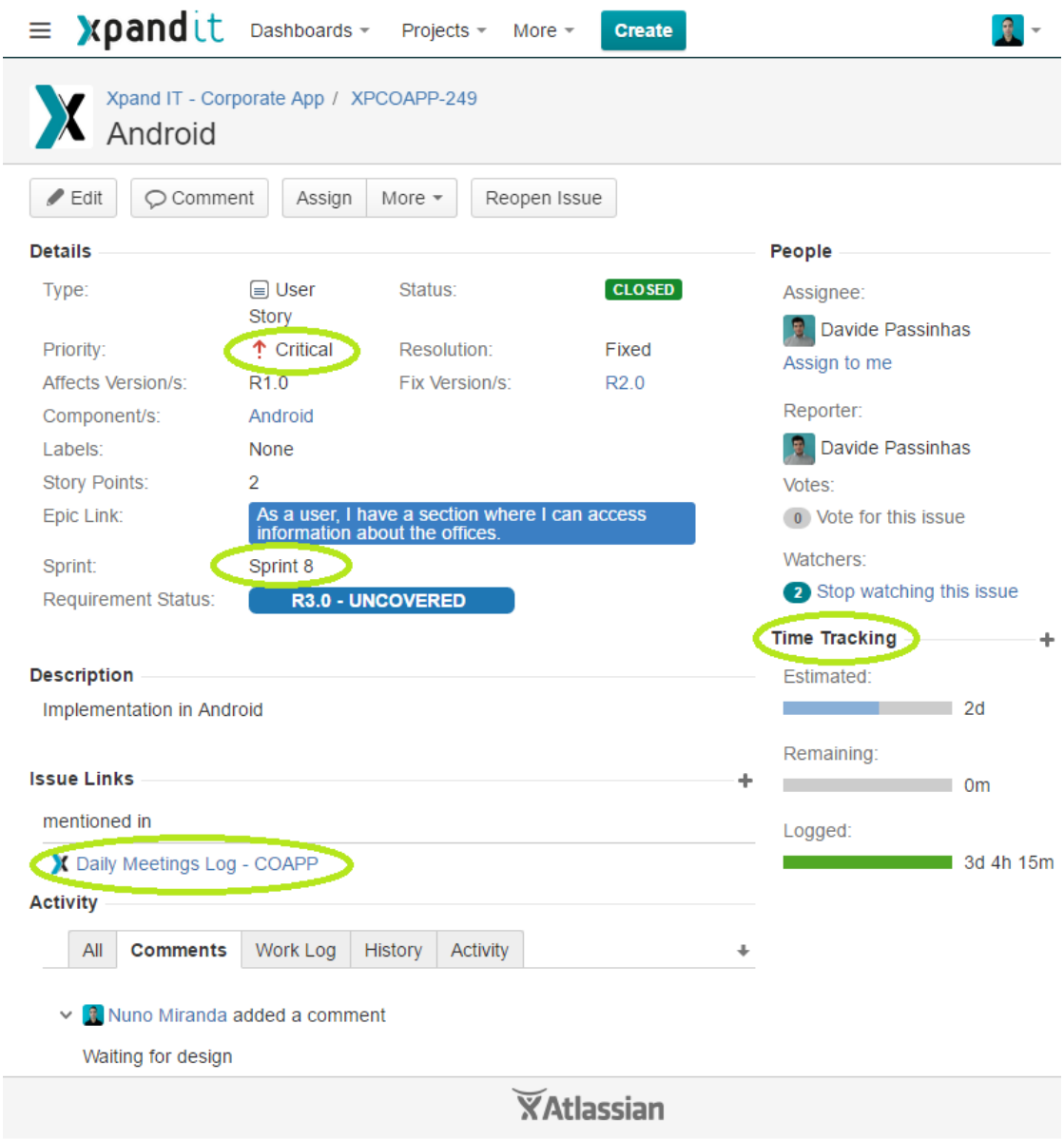

Figura 3.5: JIRA

Na próxima figura (ver Fig. 3.6) está retratado um *flow* do dia a dia da metodologia utilizada neste projeto. Tudo começa com a *"Daily Meeting"* (ver 3.1.3) onde são discutidas as tarefas que foram feitas no dia anterior e que irão ser realizadas no dia corrente; De seguida, se ainda houverem *issues* pendentes estas têm de ser completas; Quando estas estiverem completas, têm de ser marcadas como tal na plataforma JIRA e de seguida caso todas as *issues* planeadas para

aquele dia na *"Daily Meeting"* estejam completas vamos procurar por *issues* que estejam abertas ou atribuídas a mim, sempre por ordem de prioridade (*issues blocker* são resolvidas primeiro sobe *minor issues*). O ciclo repete-se até que comece um novo dia com uma nova *"Daily Meeting"*.

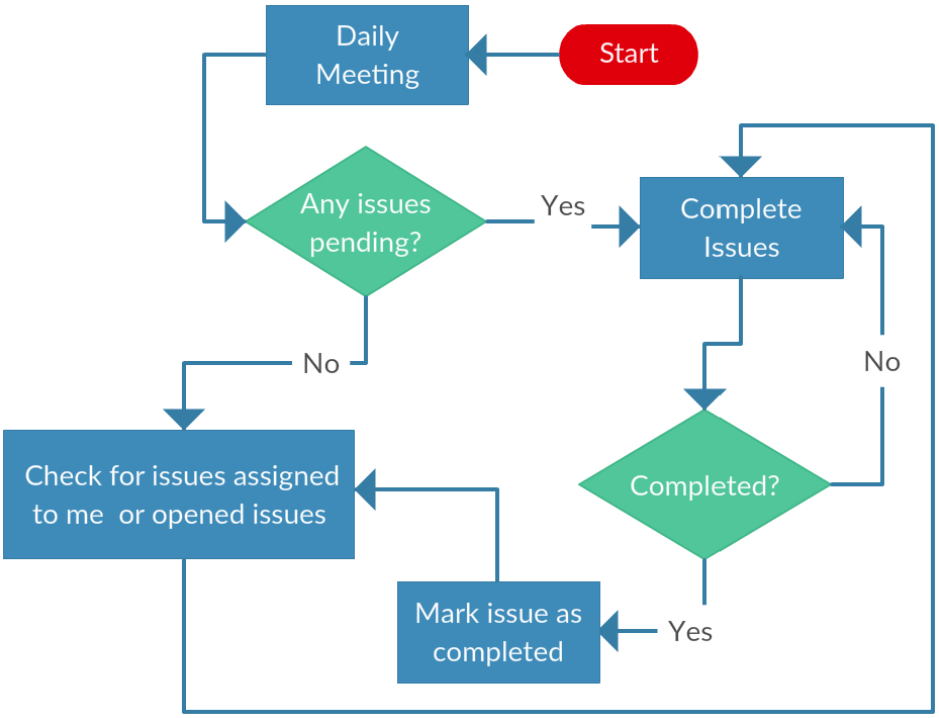

Figura 3.6: Dia a dia da metodologia

3.2.1 Product Backlog

A seguir vai ser apresentado o *product backlog* deste projeto de forma resumida e por ordem de desenvolvimento:

- CORE: Criação do Core do projeto com as ViewModels base (LoginViewModel e HomeViewModel) com acesso aos serviços (para autenticação do utilizador);
- Android, UWP: Desenvolvimento do ecrã de Login;
- Android, UWP: Desenvolvimento do sistema de navegação com menu lateral;
- CORE: Construção da WebViewModel para o carregamento das *WebViews*;

- Android, UWP: Desenvolvimento do ecrã HomePage;
- CORE: Construção do acesso aos serviços para obter a informação dos contactos de todos os colaboradores;
- CORE: Construção da EmployeesViewModel e EmployeesDetailViewModel;
- Android, UWP: Desenvolvimento do ecrã da Lista de contactos e do detalhe do contacto de um colaborador;
- Android, UWP: Desenvolvimento de um sistema de notificações para quando um colaborador comemora o seu aniversário;
- CORE: Construção da AppsViewModel;
- Android, UWP: Desenvolvimento do ecrã Apps com acesso às principais aplicações da empresa;
- CORE: Construção da OfficesViewModel;
- Android, UWP: Desenvolvimento do ecrã Offices com sistema de navegação em tabs;
- CORE: Construção da FiscalViewModel;
- Android, UWP: Desenvolvimento do ecrã Fiscal Information;
- CORE: Construção da SendContactViewModel;
- Android, UWP: Desenvolvimento do ecrã Send My Contact com sistema de navegação em tabs;
- CORE: Criação de uma cache offline para que a aplicação possa ser utilizada em modo offline;
- CORE: Criação do acesso ao serviço das Timesheets;
- CORE: Construção da TimesheetsViewModel, TimesheetEntriesViewModel e TimesheetEditEntriesViewModel;
- Android, UWP: Construção do ecrã principal das Timesheets, do ecrã de Timesheet Entries e do ecrã de adicionar/editar uma entrada;
- Serviços: construção do serviço de ausências/férias;

- CORE: Criação do acesso ao serviço das férias;
- CORE: Construção da VacationsViewModel e da VacationsMonthViewModel;
- Android, UWP: Desenvolvimento do ecrã Vacations e do ecrã VacationsMonth;
- CORE: Construção da EditViewModel e respetivos acessos aos serviços;
- Android, UWP: Desenvolvimento do ecrã de editar perfil.
- Serviços: construção do serviço de Notificações Push;
- CORE: Construção do SendPushNotification ViewModel e respetivos acessos aos serviços;
- Android, UWP: Desenvolvimento do ecrã de enviar notificação push.

Capítulo 4

Requisitos

Os requisitos são uma ferramenta essencial que permite documentar quais as funcionalidades e/ou comportamentos que a aplicação deve ter em determinadas situações e de uma forma geral. Nesta secção vão ser apresentados tantos os requisitos funcionais, como os requisitos não-funcionais da aplicação Xpack, especificando as propriedades e funções necessárias/desejáveis.

4.1 Requisitos Funcionais

Os requisitos funcionais descrevem de uma forma resumida quais as funcionalidades e serviços que o aplicação tem, devia ou poderia ter (*Must*, *Should*, *Could* – respetivamente).

Tabela 4.1: Requisitos Funcionais

ID	Descrição	Prioridade
FR01	**Login:** O utilizador tem de conseguir iniciar sessão na aplicação quando introduzidas as credenciais corretas, assim como guardar as suas credenciais para não o obrigar a introduzir sempre as credenciais assim que inicia a app. Caso as credenciais estejam incorretas, deve ser lançado um aviso a avisar tal situação.	MUST
Continua na próxima página		

Tabela 4.1 – continuação da página anterior

ID	Descrição	Prioridade
FR02	**Notícias:** O utilizador deve conseguir visualizar as últimas notícias internas da empresa num mural da página inicial, assim como ver o detalhe desta.	COULD
FR03	**Timesheets:** O utilizador tem de conseguir aceder às suas *Timesheets*, assim como tem de conseguir: • Visualizar todas as timesheets desde o momento em que entrou na empresa; • Copiar as entradas de uma semana inteira para outra semana; • Copiar apenas as entradas de um dia, de uma semana para a outra (ou mesmo dentro da mesma semana); • Adicionar novas entradas; • Guardar e/ou submeter as timesheets de uma dada semana.	MUST
FR04	**Vacations:** O utilizador tem de conseguir aceder às suas ausências, assim como: • Adicionar novas ausências ao mapa; • Verificar a quantidade de ausências ainda pode gastar por cada tipo de ausência existente; • Ao rodar o ecrã, deve visualizar um gráfico de barras anual onde consegue ver o resumo das ausências no ano presente; • Guardar as novas ausências adicionadas.	MUST
FR05	**JIRA:** O utilizador deve conseguir aceder à plataforma JIRA via web, através da Xpack.	SHOULD

Continua na próxima página

Tabela 4.1 – continuação da página anterior

ID	Descrição	Prioridade
FR06	**Confluence:** O utilizador deve conseguir aceder à plataforma Confluence via web, através da Xpack.	SHOULD
FR07	**Service Desk:** O utilizador deve conseguir aceder à plataforma do Service desk via web, através da Xpack.	SHOULD
FR08	**Editar perfil:** O utilizador deve conseguir aceder à sua informação do perfil da empresa assim como: • Alterar a sua palavra-passe; • Adicionar/alterar os seus contactos telefónicos.	SHOULD
FR09	**Lista de contactos:** O utilizador tem de conseguir aceder à lista dos contactos de todos os colaboradores da empresa assim como: • Pesquisar um contacto; • Consultar informações do contacto (*email* profissional, skype, linkedin, etc); • Adicionar o contacto à lista de contactos do dispositivo; • Enviar uma mensagem através do dispositivo; • Telefonar através do dispositivo; • Ver um ícone a identificar caso um determinado contacto faça anos nesse dia.	MUST
FR10	**Informação fiscal:** O utilizador poderia conseguir aceder à informação fiscal da Xpand IT Portugal e de Inglaterra. Ao carregar em "Portugal"deve aparecer um código de barras correspondente ao número de identificação de pessoa coletiva da Xpand IT.	COULD

Continua na próxima página

Tabela 4.1 – continuação da página anterior

ID	Descrição	Prioridade
FR11	**Morada dos escritórios:** O utilizador deve conseguir aceder à morada dos escritórios de Portugal e Inglaterra, assim como: • Ter acesso aos contactos telefónicos/fax do escritório respetivo; • Ter acesso ao mapa do escritório; • Ao carregar num botão, obter um redirecionamento para as direções para essa mesma morada através da localização atual.	SHOULD
FR12	**Notificações aniversário:** O utilizador tem de receber uma notificação quando um ou mais colaboradores fizer(em) anos.	MUST
FR13	**Lançar notificações Push:** O utilizador tem de conseguir lançar uma notificação push para todos os funcionários, mas única e exclusivamente se tiver permissões para tal (visto que não é qualquer colaborador que pode lançar uma notificação push).	MUST
FR14	**Receber notificações Push:** O utilizador tem de receber uma notificação push quando esta for lançada caso esteja com acesso à Internet, ou receber esta assim que obter um acesso à Internet.	MUST
FR15	**Enviar contacto:** O utilizador deve conseguir enviar o seu contacto por *email* ou enviar um Vcard caso tenha permissões para tal.	SHOULD
FR16	**Terminar sessão:** O utilizador tem de conseguir terminar sessão na app.	MUST
FR17	**Menu:** A aplicação tem de ter um sistema de menu lateral para conseguir aceder à maioria dos requisitos funcionais mencionados anteriormente.	MUST

Continua na próxima página

Tabela 4.1 – continuação da página anterior

ID	Descrição	Prioridade
FR18	**Offline Mode:** A aplicação deve ter um modo *offline* que permite uma utilização básica da aplicação. Isto inclui ir à lista de contactos e fazer todas as operações que permite em modo *online*, assim como deve conseguir aceder ao ecrã da morada dos escritórios e ao ecrã de informação fiscal.	MUST

4.2 Requisitos não-funcionais

Os requisitos não-funcionais definem as propriedades que o sistema deve ter ou restrições que estão implicadas no sistema. A seguir é apresentada uma lista com os requisitos não-funcionais mais importantes considerados para o desenvolvimento deste projeto.

Tabela 4.2: Requisitos não-funcionais

ID	Descrição	Prioridade
NFR01	**Interface:** A aplicação deve apresentar uma interface simples, limpa e amiga do utilizador. Isto significa que cada ecrã deve ser estudado de forma a não causar confusão ou dúvidas aquando da utilização por um utilizador inexperiente.	MUST
NFR02	**Segurança e privacidade:** A segurança e a privacidade do utilizador não devem, em qualquer circunstância, estar comprometidas. Como exemplo, todas as comunicações com servidores externos devem usar autenticação; e a palavra passe do utilizador deve ser guardada recorrendo a um algoritmo de encriptação (AES, por exemplo).	SHOULD

Continua na próxima página

Tabela 4.2 – continuação da página anterior

ID	Descrição	Prioridade
NFR03	**Lógica de negócio do código:** Tendo em conta a arquitetura organizacional do código deste projeto (ver secção 5.2.4), deve estar a maior quantidade possível de código no Core e o menos quanto possível nas plataformas respetivas. Isto permite que sejam feitas alterações muito mais rapidamente, e evita ambiguidades entre as plataformas.	MUST
NFR04	***Performance* e Responsividade:** A aplicação deve ser responsiva e dar sempre a maior quantidade de informação possível ao utilizador do que está a acontecer em *background*, isto é, sempre que existir carregamento de dados deve haver um *loading icon* para informar que algo está a acontecer. Caso exista alguma operação que o utilizador tenha feito, que tenha causado alguma alteração no sistema, este deve ser informado com um alerta. Para além disso, a aplicação não deve demorar mais tempo do que o esperado a fazer as operações, de modo a que aplicação do ponto de vista do utilizador seja fluida.	MUST
NFR05	**Metodologia de desenvolvimento:** Deve ser usada uma metodologia de desenvolvimento ágil em que todas as tarefas devem estar registadas na plataforma JIRA.	SHOULD

Capítulo 5

Desenvolvimento

Dado o calendário proposto na secção 1.3, foi desenvolvida uma primeira *release* da aplicação empresarial para uso interno dos colaboradores da Xpand IT denominada "Xpack", *release* esta correspondente à primeira fase do projeto. Dado o levantamento de requisitos feito a 25 de Março de 2016 de acordo com o referido calendário, teve como resultado a criação da seguinte lista de funcionalidades face às presentes necessidades dos colaboradores:

- Página inicial com as últimas notícias internas da empresa;
- Atalhos para as principais aplicações web da empresa;
- Lista dos contactos de cada um dos colaboradores da empresa (contacto telefónico, *mail*, etc.);
- Acesso à morada dos três escritórios da Xpand IT, com visualização em mapa;
- Informação fiscal da Xpand IT;
- Envio de contacto por Email ou Vcard;
- Menu lateral para aceder aos componentes referidos anteriormente;
- Notificação dos aniversários dos colaboradores, assim como a colocação de um *icon* de aniversário em frente ao contacto do colaborador na lista de contactos.

No fim de Setembro de 2016, período definido no calendário como a segunda fase de estágio, foi realizado o levantamento de requisitos para a segunda parte do estágio que resultou na seguinte lista de funcionalidades a desenvolver:

- **Timesheets:** cada colaborador pode registar as horas que trabalhou em cada projeto durante todos os dias de trabalho. É possível copiar as entradas de uma semana para a outra assim como copiar as entradas de um dia para o outro. No final é possível guardar ou submeter a *timesheet* daquela dada semana;
- **Mapeamento de ausências:** os colaboradores vão poder submeter o mapa das ausências, sendo possível editar e submeter a proposta das ausências assim como registar qualquer outro tipo de ausência;
- **Edição de perfil:** os colaboradores irão ter acesso ao seu perfil privado na empresa, isto é, informações pessoais e fiscais. Para além disso, vai ser possível alterar as suas credenciais, para acesso não só a esta aplicação, como também a outras aplicações *web* internas da empresa, assim como adicionar um novo contacto telefónico;
- **Segurança:** as *passwords* irão ser guardados de forma cifrada recorrendo ao algoritmo de criptografia *Advanced Encryption Standard* (AES);
- ***Scripts* de teste** definidos para a *release 1.0* e posteriormente para a *release 2.0* da Xpack, recorrendo à ferramenta Xamarin Test Cloud que permite realizar testes funcionais à aplicação em centenas de dispositivos reais.

5.1 Modelo *Model-View-ViewModel*

As aplicações em Android e Windows foram desenvolvidas segundo o modelo *Model-View-ViewModel* (MVVM) que tem a seu favor o facto de tornar a manutenção facilitada uma vez que existe uma clara separação das responsabilidades. Este modelo é composto por três componentes (ver figura 5.1):

- **View:** Esta componente é responsável pela parte visual da aplicação, ou seja, código específico da plataforma: AXML/XML no caso do Android e XAML no caso do Windows. Não é suposto haver qualquer outra responsabilidade ou regra de negócio nesta componente;
- **ViewModel:** Esta é a peça essencial no modelo MVVM, pois é responsável por fornecer à *View* a lógica de apresentação e vai coordenar as iterações da *View* com o *Model* já que estas não sabem da existência uma da outra. A *ViewModel* não tem qualquer conhecimento de como a *View* é implementada, limitando-se apenas a disponibilizar as informações e operações que esta necessita. Sempre que existe uma atualização de informação, esta é atualizada

também na *View*. Esta operação é denominada de *Data Binding* e permite que os dados na *View* estejam sempre atualizados sem qualquer intervenção por parte do programador. No caso da lista de contactos da Xpack, por exemplo, caso exista um novo colaborador na empresa, este é adicionado à base de dados. A *ViewModel*, ao detetar que houve uma alteração, automaticamente notifica a *View* e o novo colaborador aparece imediatamente na lista de contactos;

- **Model:** Nesta componente encontra-se toda a lógica de dados e de negócio, isto é: acesso a dados (neste caso, acesso aos serviços) e tratamento e validação da informação externa recebida. Sempre que existem alterações, o *Model* está responsável por notificar a *ModelView* das alterações, para que esta possa posteriormente notificar a *View*.

Figura 5.1: Modelo MVVM

5.2 Arquitetura

Nesta secção é definida a arquitetura que permite à Xpack funcionar de forma consistente.

5.2.1 Serviços externos

Como podemos ver na figura 5.2, existem vários serviços que estão a ser consumidos pela Xpack via REST, de forma a apresentar os dados necessários. São feitos vários pedidos GET e POST para obter e enviar o conteúdo necessário.

Explicando a figura, de forma geral todos os pedidos feitos pela aplicação passam primeiro por um serviço denominado de "*Load Balancer*"(exceto os pedidos com alvo o "Vacations Service"ou o "Notifications Service", serviços estes que irão ser explicados mais à frente). O "Load Balancer"está responsável por redirecionar todo o tráfego que chega aos servidores da Xpand IT para os serviços/servidores

corretos. No caso do serviço alvo for o "Jira", "Jirasd", "Confluence"ou "XP Bizcard", o percurso na figura passa apenas pelo "*Load Balancer*"que imediatamente reencaminha para os serviços respetivos. No caso do alvo ser o "Accounts Service"ou o "Xtracker Service", o percurso passa também pelo *Enterprise Service Bus* (ESB) e posteriormente pelo *Data Service Server* que irá buscar ou enviar então as informações para o "Accounts Service"ou para o "Xtracker Service".

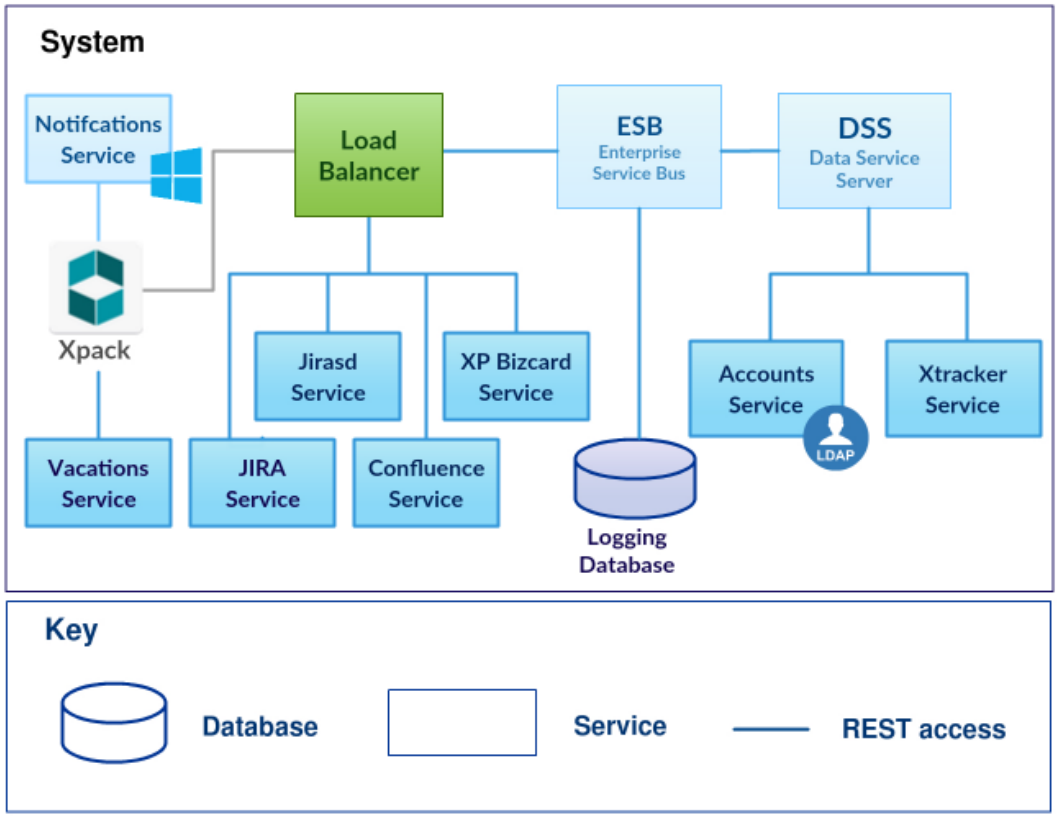

Figura 5.2: Xpack - Arquitetura

Na seguinte lista está descrito como é que cada um destes serviços é utilizado e as particularidades de cada um deles:

- **Accounts:** são feitos pedidos GET para realizar o login na Xpack (*account information*) e para obter as informações sobre todos os colaboradores da empresa (essencialmente para ser usado no ecrã de contactos); este é um serviço de diretório, normalmente conhecido por *Lightweight Directory Access Protocol* (LDAP);

- **JIRA:** para aceder ao JIRA (serviço este explicado mais detalhadamente na secção 3.1.5) através de uma *webview*;

- **Confluence:** para aceder ao Confluence (página que agrega várias informações internas à empresa) através da *webview* e pedidos GET para obter as últimas notícias da empresa;

- **Xp Bizcard:** serviço utilizado apenas para envio de Vcard, através de pedidos POST;

- **JIRASD:** serviço usado para aceder ao Service Desk através de uma *webview*.

- **Vacations Service:** este serviço tem como objetivo fornecer os dados das ausências dos colaboradores e/ou submeter novos dados; Este serviço está à parte na figura, e é acedido diretamente pela Xpack visto que está alojado num serviço construido completamente à parte de todos os outros serviços. Isto acontece porque este é apenas um serviço que está em fase de testes (visto que o sistema de ausências está a sofrer uma grande reformulação e ainda não está completo para ficar disponível ao público alvo em geral) e que está alojado localmente nas instalações da Xpand IT – é acedido apenas dentro da rede interna da empresa (através de uma VPN a partir do exterior). Detalhes de implementação sobre este serviço estão disponíveis na secção 5.4.2.1;

- **Notifications Service:** este serviço tem como objetivo o lançamento de notificações push para os dispositivos Android e Windows. Este serviço foi publicado no Azure, estando assim a funcionar na *cloud*. Mais detalhes de implementação sobre este serviço estão disponíveis na secção 5.4.3;

- **Xtracker Service:** este serviço pretende dar resposta a qualquer tipo de pedidos que sejam feitos a partir da funcionalidade das Timesheets;

- **ESB:** este serviço serve como reencaminhador de determinado tipo de pedidos para esta aplicação e faz *logging* de todos os pedidos que passem por este numa base de dados. Este serviço apesar de não ser propriamente necessário aos olhos deste diagrama, existem muitos outros serviços que estão a ser ligados a este ESB, no entanto não são utilizados por esta aplicação, não estando assim presentes no diagrama;

- **DSS:** este serviço tem como função facilitar a gestão e o acesso à informação.

Para fazer o acesso a cada um destes serviços é necessária a autenticação através do *HTTP Basic Authenticator*. É importante salientar que todos os serviços aqui representados (ver figura 5.2) já se encontravam desenvolvidos pela empresa, estando fora do âmbito de desenvolvimento desta dissertação (com exceção dos serviços "*Vacations Service*"e "*Notifications Service*"que foram desenvolvidos no contexto desta dissertação). Estes serviços estão alojados de diferentes formas, estando alguns espalhados pela *cloud* (*e.g.* Azure[1]) e outros em servidores locais como já foi referido.

5.2.2 Serviços internos

Os serviços internos são, como o nome indica, serviços que funcionam dentro da aplicação Xpack. Estes são classes pertencentes ao *Core* (ver secção 5.2.4) que estão responsáveis por efetuar a comunicação com os diversos serviços externos referidos anteriormente e essenciais para que a aplicação funcione no seu pleno.

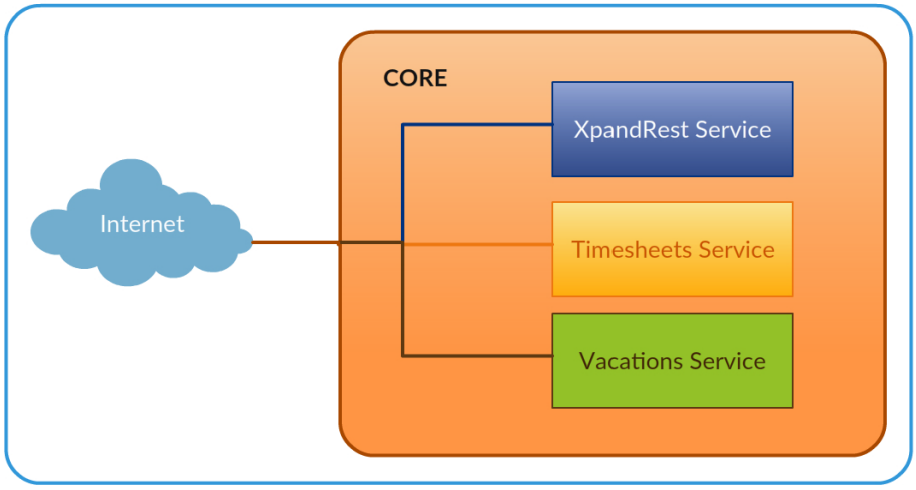

Figura 5.3: Xpack - Serviços Internos

Estão assim subdivididos em três principais serviços:

- **XpandRest Service:** responsável pela recolha de informação de todos os colaboradores para: a lista de contactos; informação fiscal da empresa;

[1]Plataforma destinada à execução de aplicações e serviços, baseada nos conceitos da computação em nuvem (*cloud*), URL: `https://azure.microsoft.com/`

notícias; acesso ao Confluence, JIRA e JIRA Service Desk; informação de perfil, incluindo as alterações de credenciais do utilizador; e informação da morada dos escritórios da empresa;

- **Timesheets Service:** responsável por todas as operações relativas às Timesheets, nomeadamente a recolha das das timesheets mais recentes, assim como a submissão das mesmas;

- **Vacations Service:** responsável por todas as operações relacionadas com o mapa de ausências, nomeadamente: o fornecimento do estado atual do mapa de férias do utilizador correspondente; o fornecimento dos feriados do respetivo ano; estatísticas relativas ao ano selecionado; e a submissão das alterações efetuadas para posterior aprovação.

Para além destes três principais serviços internos (todos eles no Core), temos alguns outros serviços secundários que por vezes recorrem aos serviços a cima mencionados:

- **Local Storage Service:** Este serviço tem como função guardar os dados básicos da aplicação (lista dos colaboradores, *feed* de notícias e lista dos aniversários) na *storage* do dispositivo para que estas informações possam ser acedidas em modo *offline*;

- **Birthday Service:** Este serviço tem como função calcular e providenciar a lista dos colaboradores que festejam o seu aniversário; este serviço recorre ao serviço anterior para ir buscar a lista dos contactos (e seus aniversários) para evitar tempos de espera (no caso de ir buscar os contactos aos serviços externos);

- **Clipboard Service:** Este serviço foi criado com o objetivo de dar apoio a uma função da funcionalidade Timesheets: copiar e colar as entradas de um dia para o outro (para mais informações ver secção 5.4.1);

- **Cryptography Service:** Este serviço tem a função de encriptar e desencriptar a palavra passe do utilizador; Os dados são encriptados com AES recorrendo a uma biblioteca – PCLCrypto[2].

[2]Para mais informações consultar: `https://github.com/aarnott/pclcrypto` (acedido em Dezembro de 2016)

5.2.3 Local Storage

Para além disso, existe uma outra funcionalidade que exige o acesso à *local storage* do dispositivo móvel. Esta funcionalidade é denominada *offline mode* e permite utilizar a aplicação mesmo que não exista acesso à Internet. No entanto, será necessário ter o login feito na Xpack, pelo menos uma vez com acesso à Internet de forma a que estes dados sejam guardados na *local storage*. Desta forma são guardados dados que normalmente só estão disponíveis *online*, isto é: acesso à aplicação (login), dados dos contactos dos colaboradores e resumo das notícias. Estão excluídos do modo *offline* o acesso ao JIRA, Confluence e Service Desk já que é mesmo necessário interagir com os serviços para carregar a *webview*. O mesmo se aplica ao acesso da funcionalidade de Timesheets, assim como à funcionalidade do mapa de ausências (isto é, é possível aceder aos conteúdos desde que já tenham sido carregados uma vez, mas não é permitido fazer alterações).

5.2.4 Modelo MVVM aplicado ao projeto

Na figura 5.4 está retratada a arquitetura da Xpack do ponto de vista de programação. Esta é constituída por três projetos diferentes: o projeto Android, o projeto Windows e o projeto Core que agrega toda a lógica de negócio e incorpora todo o código que é comum às três plataformas. Este processo é explicado melhor na figura seguinte (ver figura 5.5).

Se tentarmos encaixar o diagrama da Figura 5.1 (ilustração do modelo MVVM) com o diagrama da Figura 5.4 poderíamos obter facilmente algo como a Figura 5.5. No Core é onde a componente *ViewModel* e *Model* estão representadas. Neste caso a componente *Model* está responsável por comunicar com os serviços tal e qual como está ilustrado na figura. O projeto Android e Windows apenas correspondem à componente *View*. Este conceito representa os principais valores da Xamarin, onde todo o código essencial e comum às três plataformas, neste caso, duas plataformas, se encontra agregado apenas num projeto.

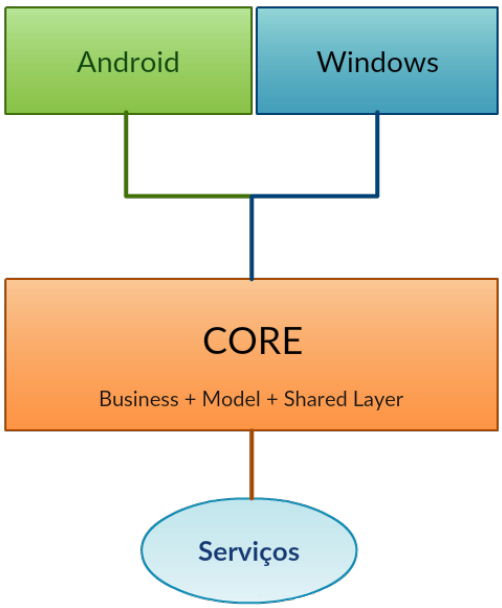

Figura 5.4: Arquitetura Xamarin

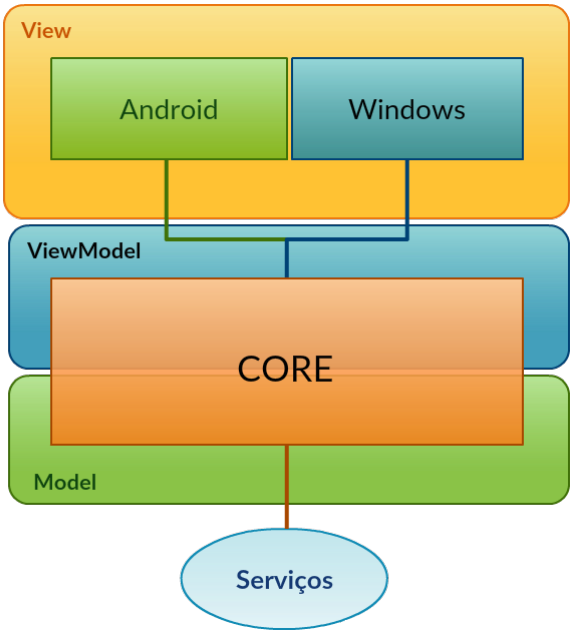

Figura 5.5: Arquitetura Xamarin com modelo MVVM

5.2.4.1 *ViewModels*

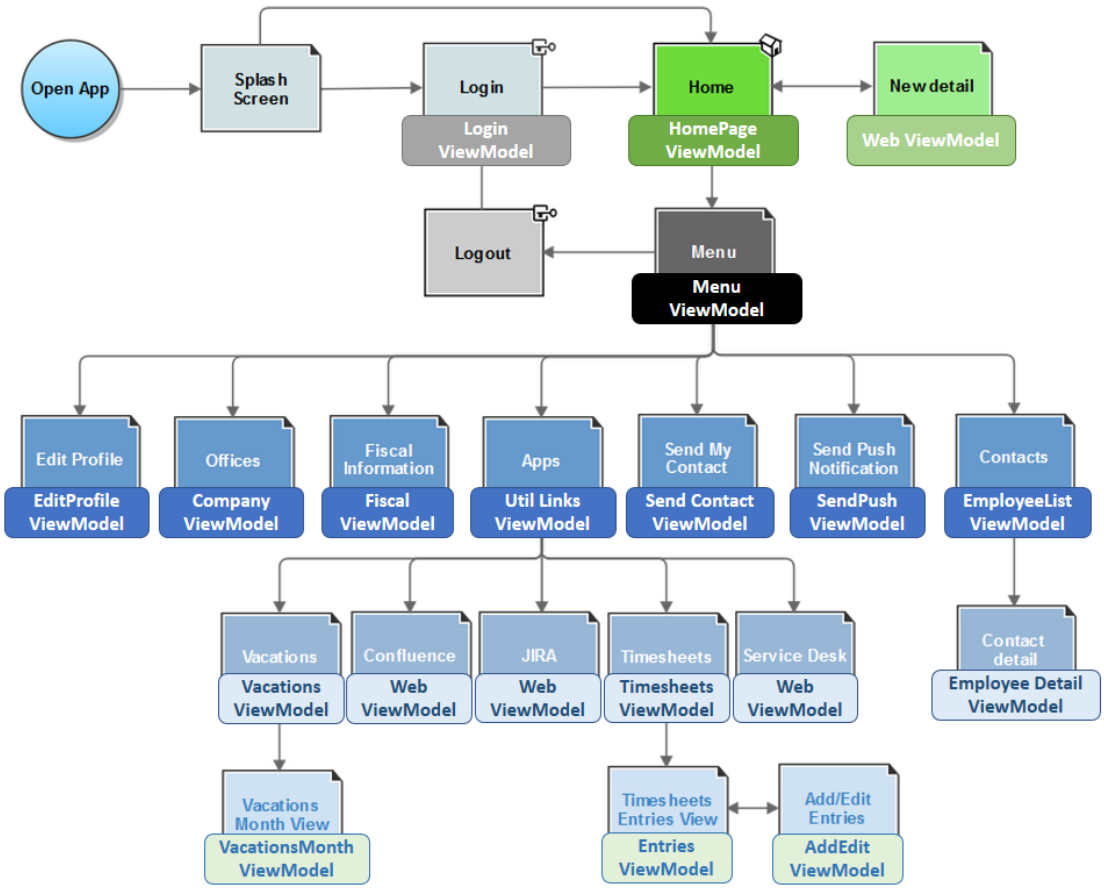

Figura 5.6: Xpack - Navegação entre as *ViewModels*

Na figura 5.6 é apresentada a forma está a ser feita a navegação entre as diversas *ViewModels*), isto é:

- **Login ViewModel** – esta *ViewModel* corresponde ao ecrã de login e está responsável por:
  - Fazer as operações de autenticação, isto é, verificar se existe conexão de Internet. Caso exista, recorre aos serviços e verifica se as credenciais estão corretas ou não. Se não existir conexão, verifica a *local storage* para validar se existem as credenciais em modo *offline*;
  - Guardar as credenciais ao fazer *check* na *label* "Remember";

  - Apresentar alerta de erro caso as credenciais estejam incorretas;
  - Encaminhar para a "Home ViewModel"caso as credenciais estejam válidas.

- **Home ViewModel** – esta é a primeira *ViewModel* a ser carregada após a autenticação ser efetuada com sucesso e está responsável por diversas operações:
  - Apresentar o ecrã de autenticação caso seja verificado que a mesma não está efetuada;
  - Se a operação anterior não se verificar então são carregadas duas *ViewModels*: a Homepage ViewModel e a Menu ViewModel;
  - Responsável pela gestão dos comandos que aparecem na *action bar*: "Go to homepage"e "Go to contacts";
  - Processa os aniversários – verifica se existe algum colaborador que festeje o seu aniversário no dia atual. Caso exista, lança uma notificação. Esta notificação é apenas apresentada se a Xpack nunca tiver sido aberta no dia atual;
  - Guarda informações na *local storage* para o modo *offline* funcionar, entre elas: lista dos colaboradores, com os seus contactos; lista das últimas cinco notícias; e a informação de autenticação.
- **Menu ViewModel** – *ViewModel* responsável por todas as ações do Menu:
  - *Resources* de todos os textos apresentados do menu;
  - Ações de todos os botões, isto é, os botões para navegar pelos restantes ecrãs do menu que reencaminham para as correspondentes *ViewModels*;
  - Responsável pelo logout na aplicação: remove as credenciais e mostra a *Login ViewModel*.
- **HomePage ViewModel** – esta *ViewModel* corresponde ao ecrã da HomePage e está responsável por:
  - Fornecer as hiperligações do JIRA e JIRA Service Desk (que são fornecidas pelos serviços internos) para carregar a *WebView* que é ativada ao carregar nos botões respetivos;
  - Fornecer uma lista com as últimas notícias internas da empresa e guardar as mesma numa *local storage*, de forma a que possam ser visualizados quando o dispositivo móvel se encontra em modo *offline*. Esta lista é

armazenada numa *Observable Collection*, pois permite que a mesma fique sempre atualizada, caso existam alterações na informação devolvida pelos serviços (ver *binds* na secção 5.1);

- Fornecer *resources* textuais que existam no ecrã, por exemplo, os textos "Home", "Company News", "TIMESHEETS", "JIRA"e "SERVICE DESK";
- Ação dos dois botões das aplicações: Confluence e JIRA Service Desk, que vai ativar outra *ViewModel* para carregar a *WebView*; assim como a ação do botão "Timesheets"que irá carregar a respetiva*ViewModel*.

- **Web ViewModel** – esta *ViewModel* é ativada quando é chamada uma WebView e está responsável por:
  - Carregar a *WebView*;
  - Disponibilizar uma mensagem de aviso caso não exista conexão à Internet ou outro erro no carregamento da *WebView*;
  - Ação dos botões, por exemplo, voltar ao ecrã ao carregar no botão "Back".

- **Employee List ViewModel** – esta *ViewModel* corresponde à lista de contactos dos colaboradores da empresa e está responsável por:
  - Carregar a lista dos contactos a partir dos serviços, guardar os mesmos na *local storage* e fornecer a lista *Observable Collection* às plataformas;
  - Ação de pesquisar por um contacto, isto é, filtro de texto livre que permite filtrar os contactos pelo texto que é passado para a *ViewModel*. A lista é então atualizada para os contactos filtrados;
  - Ação de carregar num contacto, reencaminhando o utilizador para um novo ecrã com o detalhe desse contacto, o que corresponde a uma nova *ViewModel*;
  - Ação de todos os botões existentes no ecrã para: voltar ao ecrã anterior; pesquisar um contacto; carregar o detalhe de um contacto; fechar a procura; ativar a pesquisa por voz; e fazer *refresh* à lista;

- **Employee Detail ViewModel** – esta *ViewModel* é ativada quando se carrega num contacto da lista dos contactos e é mostrado o detalhe do mesmo. Esta *ViewModel* está também responsável por:

  - Disponibilizar toda a informação sobre este colaborador: nome, número de telefone, email, skype, etc.;
  - Ações de todos os botões: ação de realizar telefonema através do dispositivo; enviar uma SMS; adicionar contacto à lista de contactos do dispositivo; enviar email; abrir a aplicação "Skype"e adicionar o contacto que foi selecionado; e voltar ao ecrã anterior.

- **Util Links ViewModel** – esta *ViewModel* corresponde ao ecrã de "Apps". Está responsável pelas mesmas ações da *Homepage ViewModel* com exceção do carregamento das notícias e, com o extra de providenciar o carregamento das TimesheetsViewModel, assim como as VacationsViewModel;

- **Company ViewModel** – esta *ViewModel* corresponde ao ecrã "Offices"e está responsável por:

  - Ação dos botões "Tab"que atualizam as informações dos escritórios, bem como as coordenadas do mapa;
  - Outras ações como: realizar uma chamada ao clicar no texto com a informação do número de telefone do escritório; ou voltar ao ecrã anterior carregando no botão "Back";

- **Fiscal ViewModel** – esta *ViewModel* corresponde ao ecrã "Fiscal Information"e está responsável por:

  - Disponibilização dos *resources* de texto apresentados no ecrã;
  - Ação do botão "Go NIF card"que encaminha para outra *ViewModel*;
  - Ação do botão "Back"para voltar ao ecrã anterior.

- **NIF Card ViewModel** – esta *ViewModel* é ativada ao carregar no botão "Go Nif Card"do ecrã "Fiscal Information"e está responsável por:

  - Disponibilizar os *resources* dos textos (entre eles o número de contribuinte que é apresentado no ecrã;
  - Ação do botão "Back"que volta ao ecrã anterior.

- **Send Contact ViewModel** – esta *ViewModel* corresponde ao ecrã *"Send My Contact"* e está responsável por:

  - Enviar um *Vcard* utilizando os serviços (*"xp bizcard services"*);
  - Enviar um *email* com as informações de contacto do colaborador;

  - Trocar as informações do ecrã consoante a *tab* que está selecionada;
  - Verificar a segurança relativamente ao *email* inserido, recorrendo a expressões regulares;
  - *Resources* de todos os textos do ecrã, inclusive do texto a ser inserido no *email* de envio;
  - Apresentar um alerta, caso o *Vcard* não tenha sido enviado com sucesso ou as configurações do *email* do dispositivo não estejam corretas. Apenas acontece quando é enviado um *email*.

- **Timesheets ViewModel** – esta *ViewModel* corresponde ao ecrã principal das Timesheets que está responsável por:
  - Interagir com os serviços internos da aplicação que por sua vez vai interagir com os serviços externos à Xpack;
  - Fornecer o estado atual da *timesheet* na semana atual;
  - Fornecer a lista das entradas daquela determinada semana;
  - Carregar a semana anterior ou seguinte;
  - Encaminhar para a EntriesViewModel com os dados correspondentes aquele dia, ao carregar num dia em específico;
  - Copiar e colar um determinado dia;
  - Copiar uma determinada semana para a semana selecionada;
  - Tratar de todas as operações que estejam subjacentes às funções mencionadas anteriormente.

- **Entries ViewModel** – esta *ViewModel* corresponde ao ecrã das Timesheets, mais concretamente da listagem das *entries* de um determinado dia. Esta *ViewModel* está responsável por:
  - Receber o dia selecionado com as respetivas entradas;
  - Fornecer a lista das entradas desse dia;
  - Fornecer a ação de quando uma entrada é selecionada. Esta ação deve reencaminhar para a *AddEditViewModel* com as informações correspondentes a essa entrada;
  - Fornecer a ação de quando é carregado num botão para adicionar uma nova entrada. Esta ação deve reencaminhar novamente para a *AddEditViewModel* mas sem dados subjacentes;

  - Comunicar, de forma semelhante à *TimesheetViewModel*, com os serviços internos da Xpack;
  - Todas as operações que estejam subjacentes às funções mencionadas anteriormente.

- **Add/Edit ViewModel** – esta *ViewModel* corresponde ao ecrã das Timesheets, mais concretamente da função de editar uma entrada de um determinado dia, ou adicionar uma nova entrada. Esta *ViewModel* está responsável por:
  - Interagir com os serviços externos de forma a obter a listagem dos projetos, das tarefas e dos *cost centers* do colaborador com sessão iniciada;
  - Verificar se a entrada atual já foi submetida ou não. Caso já esteja submetida não é possível editar a mesma, assim como não é possível adicionar uma nova entrada a uma semana que já esteja atualmente submetida;
  - Adicionar/Editar a entrada selecionada;
  - Tratar de todas as operações que estejam subjacentes às funções mencionadas anteriormente.

- **Vacations ViewModel** – esta *ViewModel* corresponde ao ecrã das Vacations. Neste ecrã é possível ver um gráfico de ausências do ano selecionado e escolher o mês para marcar novas ausências. Esta *ViewModel* está responsável por:
  - Apresentar as ausências do ano selecionado em forma de estatísticas;
  - Comunicação com o serviço interno da Xpack para obter a informação das ausências disponibilizadas pelo serviço externo;
  - Apresentar todos os *resources* textuais apresentados no ecrã, inclusive a informação do ano selecionado;
  - Tratar da ação de mudança de ano;
  - Tratar da ação de mudança de *ViewModel*, selecionando um dado mês para a *VacationsMonthViewModel*;
  - Gerir todas as operações que estejam subjacentes às funções mencionadas anteriormente.

- **VacationsMonth ViewModel** – esta *ViewModel* corresponde ao ecrã Calendar onde é possível fazer a marcação das ausências com vista por mês. Esta *ViewModel* está responsável por:

  - Providenciar as ausências atuais, recorrendo aos respetivos serviços;
  - Calcular o número total de cada tipo de ausência, assim como calcular o número de ausências ainda por gozar;
  - Gerir todas as operações que estejam subjacentes às funções mencionadas anteriormente.

- **Edit Profile ViewModel** – esta *ViewModel* corresponde ao ecrã de edição de perfil onde é possível, não só ver todas as informações do colaborador (tais como informações fiscais, informações bancárias, informações pessoais, entre outros), como permitir ao utilizador que altere as suas credenciais para acesso interno na empresa, ou acrescentar um número de telefone aos atuais. Esta *ViewModel* está responsável por:
  - Disponibilizar todas as informações do colaborador para serem apresentadas no ecrã, recorrendo aos serviços para ir buscar estas mesmas informações;
  - Fazer a alteração das credenciais do colaborador assim que esta tarefa for requisitada pela plataforma respetiva;
  - Acrescentar um novo número de telefone, enviando para o servidor o novo contacto.
- **Send Push ViewModel** – esta *ViewModel* corresponde ao ecrã de enviar uma notificação push. Está responsável por chamar o serviço das notificações push que consequentemente vão lançar uma notificação para todos os dispositivos Android e Windows (para mais informações, consultar secção 5.4.3). É de notar que este ecrã tem como restrições ter sessão iniciada com credenciais de administrador. Caso não tenha estas credenciais, este ecrã deixa de estar acessível no Menu da aplicação.

5.3 Especificação de alto nível

O desenvolvimento desta aplicação apenas incluiu a componente de *front-end*, já que os serviços de *back-end* já se encontravam desenvolvidos, com exceção de alguns serviços que irão ser vistos mais à frente nesta dissertação. Ou seja, foi desenvolvido tudo a nível da aplicação Android e Windows, mas a componente dos servidores onde se encontra alojada a informação sobre os colaboradores, informação sobre a Xpand IT, etc. já se encontrava desenvolvido, saindo do escopo desta dissertação.

Apesar desta metodologia valorizar mais o *software* funcional em detrimento de uma documentação exaustiva, visto que esta tese está a ser desenvolvida no âmbito de um estágio de Engenharia de Software, a documentação da Xpack foi à mesma complementada, de forma a facilitar a compreensão da mesma.

Na figura 5.7 está retratado o mapa da aplicação, onde é apresentada a sequência dos ecrãs da Xpack.

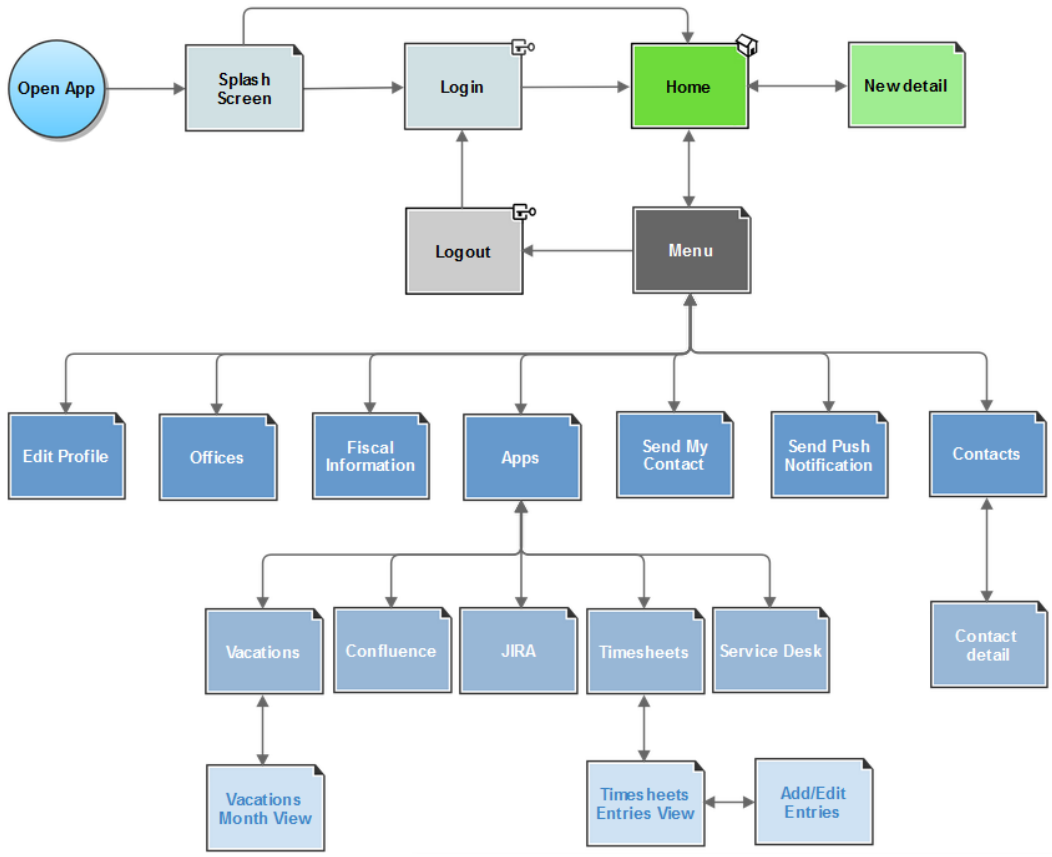

Figura 5.7: Xpack - App Map

5.3.1 User Stories

Nesta secção são descritas as *User Stories* que foram desenvolvidas para o projeto (Tabela 5.1). Estas descrevem a necessidade concisa do utilizador no produto (um requisito) sobre o ponto de vista deste. Foram estabelecidas prioridades que tiveram impacto no desenvolvimento destas funcionalidades, isto é, as *user stories* com maior prioridade foram desenvolvidas em primeiro lugar.

Tabela 5.1: User Stories

ID	Descrição	Prioridade (1-5)
US01	Como utilizador, pretendo fazer a autenticação na Xpack para que consiga aceder às funcionalidades desta.	5
US02	Como utilizador, pretendo aceder às principais aplicações internas da empresa: JIRA, Confluence e JIRA Service Desk.	4
US03	Como utilizador, pretendo ter acesso às últimas notícias internas na Xpand IT.	4
US04	Como utilizador, pretendo ter acesso aos contactos de todos os colaboradores da empresa.	5
US05	Como utilizador, pretendo receber uma notificação de quando um ou mais colaboradores festejam o seu aniversário.	3
US06	Como utilizador, pretendo ver um ícone de bolo à frente de um dado colaborador quando o mesmo faz anos.	2
US07	Como utilizador, pretendo adicionar um ou mais colaboradores à minha lista de contactos do telemóvel.	3
US08	Como utilizador, pretendo pesquisar qualquer colaborador e aceder ao seu contacto para que seja mais fácil encontrar o contacto de um dado colaborador.	3
US09	Como utilizador, pretendo aceder à localização dos 3 escritórios da empresa através de um mapa.	5
US10	Como utilizador, pretendo ter acesso aos dados fiscais da Xpand IT em Portugal e na Inglaterra.	3
US11	Como utilizador, pretendo enviar um email com a minha informação de contacto.	3
US12	Como utilizador, pretendo enviar um *Vcard*.	3
US13	Como utilizador, pretendo consultar as minhas timesheets.	5
US14	Como utilizador, pretendo adicionar novas entradas às minhas timesheets assim como devo puder editalas caso estas ainda não estejam submetidas.	5

Continua na próxima página

Tabela 5.1 – continuação da página anterior

ID	Descrição	Prioridade (1-5)
US15	Como utilizador, pretendo copiar uma semana ou um dia nas minhas timesheets para outra semana ou dia, respetivamente.	4
US16	Como utilizador, pretendo guardar ou submeter as alterações que fiz a uma determinada semana nas minhas timesheets.	5
US17	Como utilizador, pretendo consultar o meu mapa de ausências, navegando pelos diferentes anos.	5
US18	Como utilizador, pretendo adicionar novas ausências ao meu mapa.	5
US19	Como utilizador, pretendo submeter as novas ausências que introduzi no meu mapa de férias.	5
US20	Como utilizador, pretendo saber quantos ausências ainda tenho por gozar dentro da categoria de cada uma das ausências.	4
US21	Como utilizador, pretendo ter acesso ao meu perfil na empresa onde posso consultar todas as minhas informações pessoais e fiscais.	4
US22	Como utilizador, pretendo alterar as minhas credenciais de acesso dentro da empresa.	4
US23	Como utilizador, pretendo adicionar novos contactos telefónicos ao meu perfil.	3
US24	Como utilizador, pretendo terminar sessão na aplicação.	5
US25	Como utilizador, pretendo aceder aos conteúdos da aplicação em modo *offline*.	3

5.3.2 Casos de uso

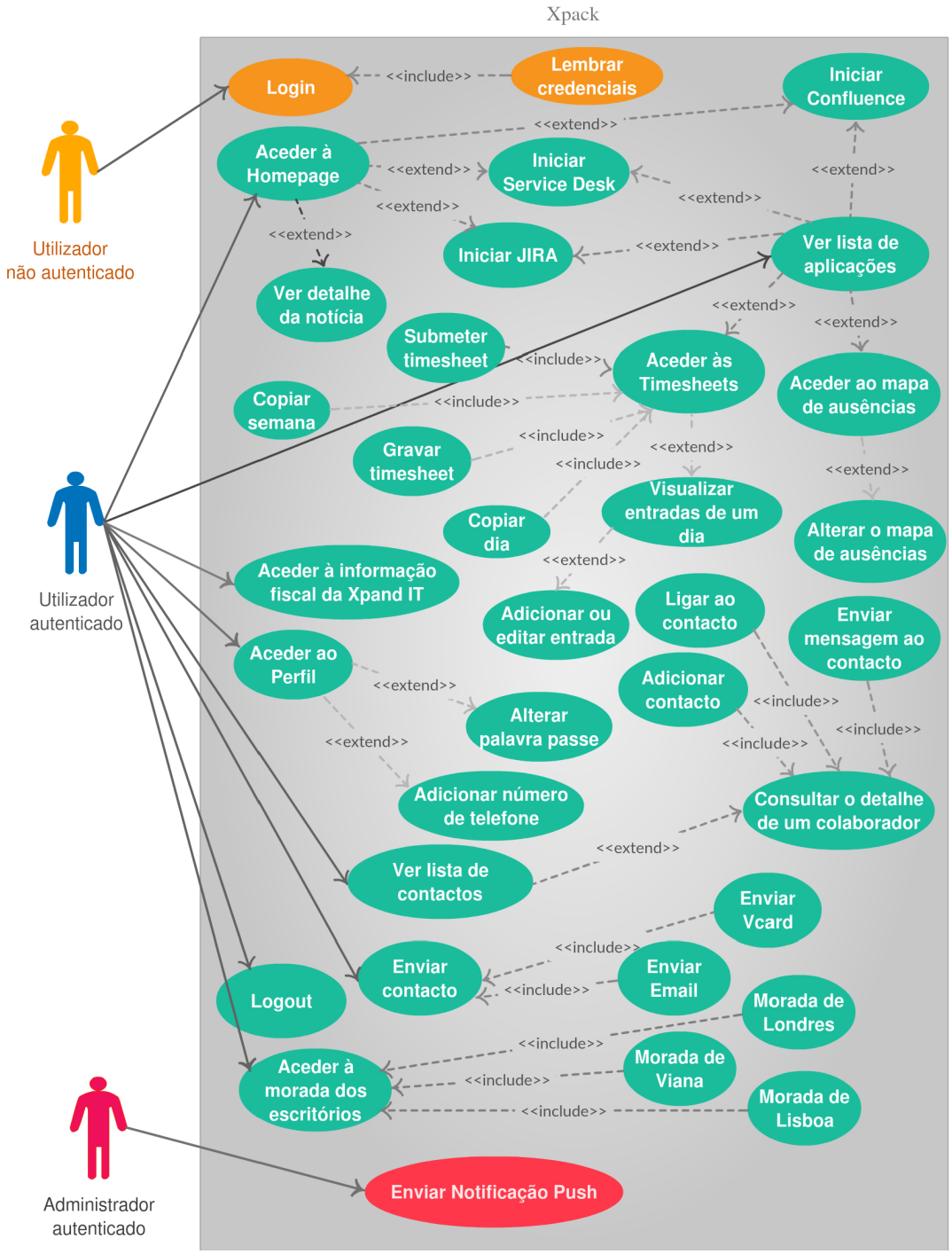

Figura 5.8: Diagrama de casos de uso

Os casos de uso são mais uma forma de mostrar os requisitos funcionais da aplicação e, embora sejam parecidos às *user stories*, um elemento não substitui o outro. Na figura 5.8 está retratado o diagrama de casos de uso e a seguir as tabelas correspondentes a cada caso de uso. Apenas foram colocadas nesta secção os casos de uso mais importantes (para ver a lista completa consultar Anexo A).

Ator	Colaborador
Prioridade	Must
Pré-condições	• Não ter sessão iniciada
Pós-condições	• Login efetuado; • É possível usufruir das funcionalidades da Xpack.

Tabela 5.2: UC01 - Login

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado.
Pós-condições	• É possível visualizar a lista de notícias; • É possível ver o detalhe de uma notícia.

Tabela 5.3: UC03 - Aceder à Homepage

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado.
Pós-condições	• É possível aceder às aplicações JIRA, Confluence e Service Desk.

Tabela 5.4: UC08 - Ver lista de aplicações

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado.
Pós-condições	• É possível aceder à lista dos contactos de todos os colaboradores da empresa.

Tabela 5.5: UC09 - Ver lista de contactos

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado; • Estar no ecrã de detalhe de um contacto.
Pós-condições	• É possível enviar uma mensagem de texto através do telemóvel ao colaborador.

Tabela 5.6: UC13 - Enviar mensagem ao contacto

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado.
Pós-condições	• É possível aceder à informação fiscal da Xpand IT em Portugal e em Inglaterra.

Tabela 5.7: UC14 - Aceder à informação fiscal da Xpand IT

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado.
Pós-condições	• É possível enviar a informação dos contactos do colaborador autenticado para outro contacto.

Tabela 5.8: UC15 - Enviar contacto

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado.
Pós-condições	• É possível aceder à morada dos três escritórios da Xpand IT.

Tabela 5.9: UC18 - Aceder à morada dos escritórios

5.3.3 Mockups

Sendo que a empresa queria um *desgin* da aplicação mais refinado e final, os *mockups* foram feitos em conjunto com a equipa de *design* permitindo um melhor *feedback* entre o que é possível fazer, o que estava planeado de acordo com o plano de requisitos e a intenção dos *designers*. Nesta secção é apresentado um conjunto de imagens onde são retratados os *mockups* mais relevantes da aplicação. Estes *mockups* foram feitos após a análise de requisitos, mas naturalmente antes do desenvolvimento da Xpack. É de notar portanto que este *design* é apenas um *mockup* inicial e sofreu alterações na aplicação final durante o desenvolvimento do projeto, com o intuito de melhorar a usabilidade. O design final está visível no final desta tese, no Anexo B.

Figura 5.9: Xpack - Splash Screen

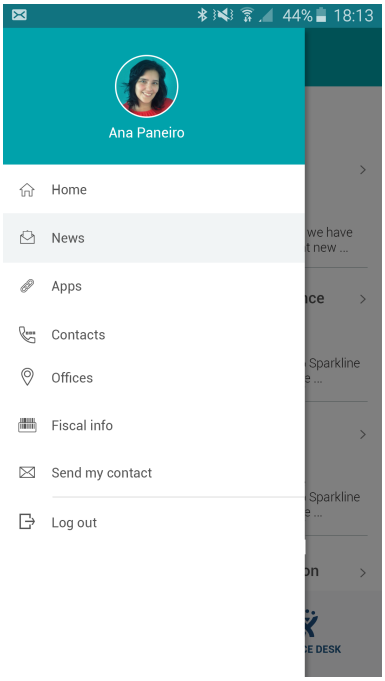

Figura 5.10: Xpack - Menu

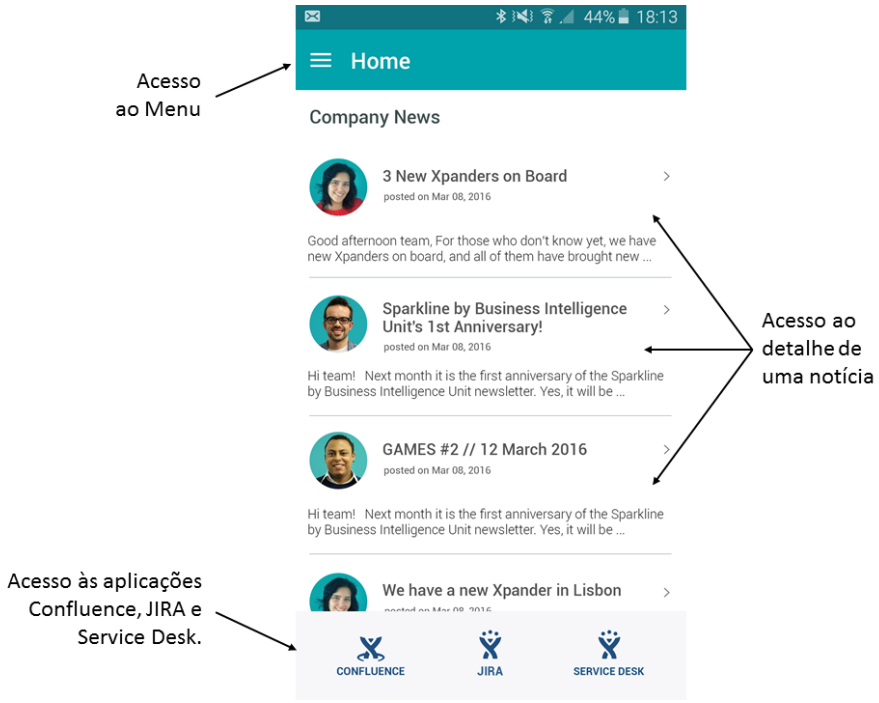

Figura 5.11: Xpack - Home

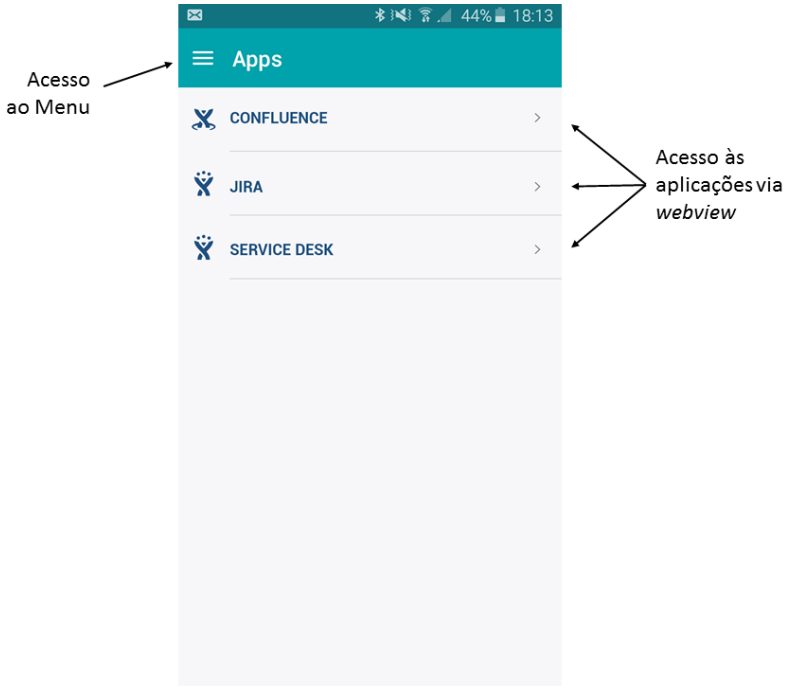

Figura 5.12: Xpack - Apps

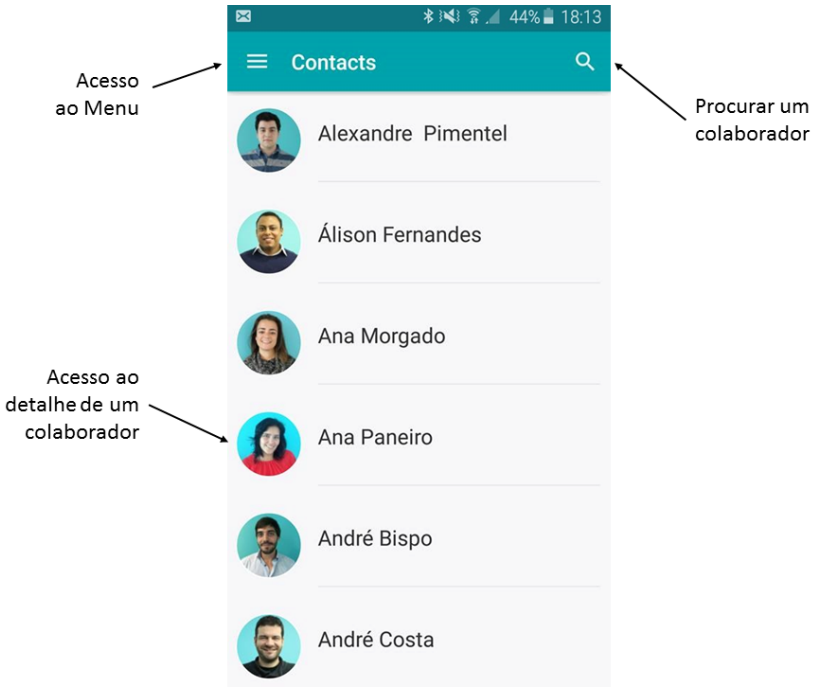

Figura 5.13: Xpack - Contactos

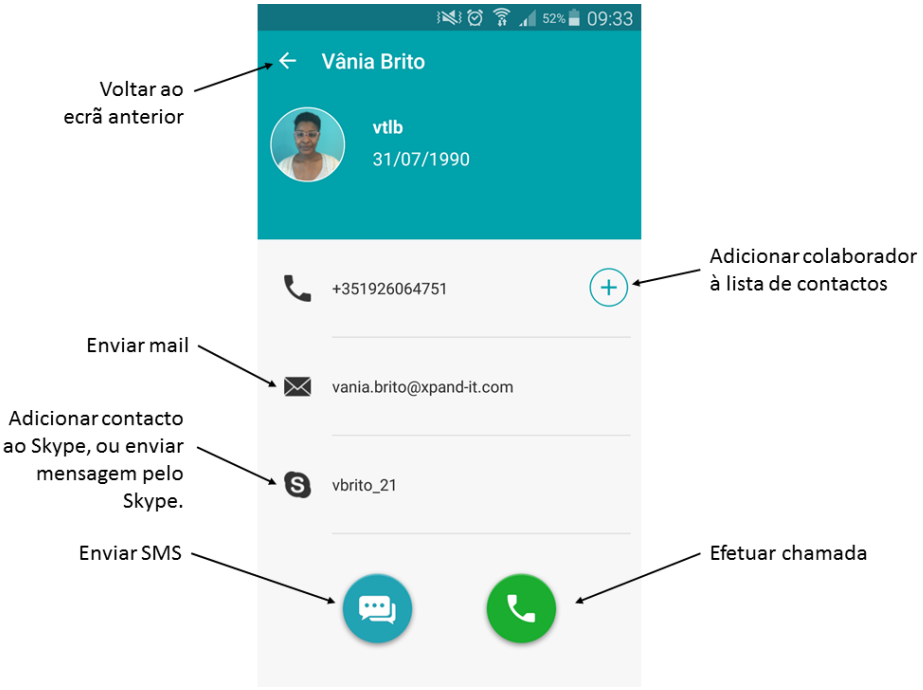

Figura 5.14: Xpack - Contactos (detalhe de contacto)

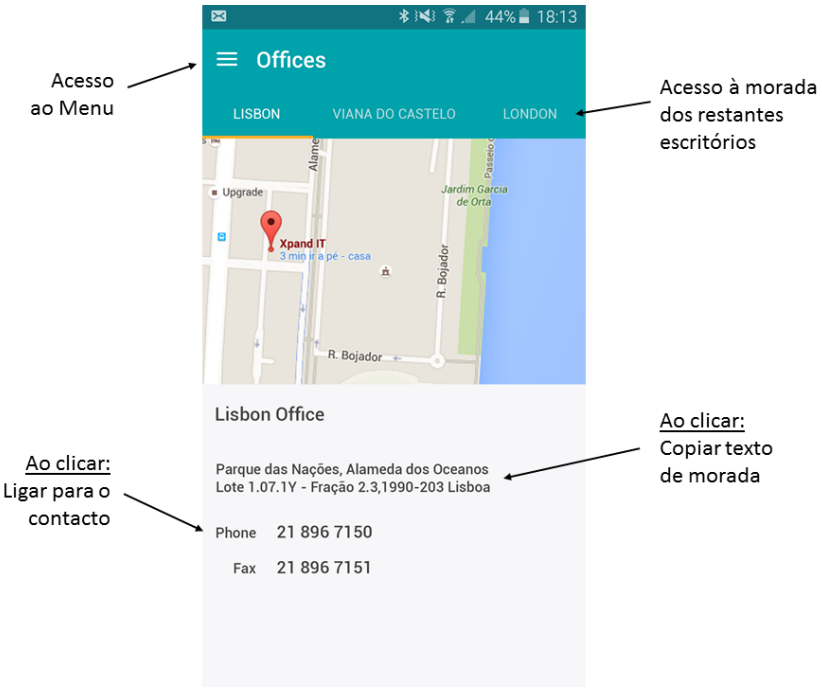

Figura 5.15: Xpack - Escritórios

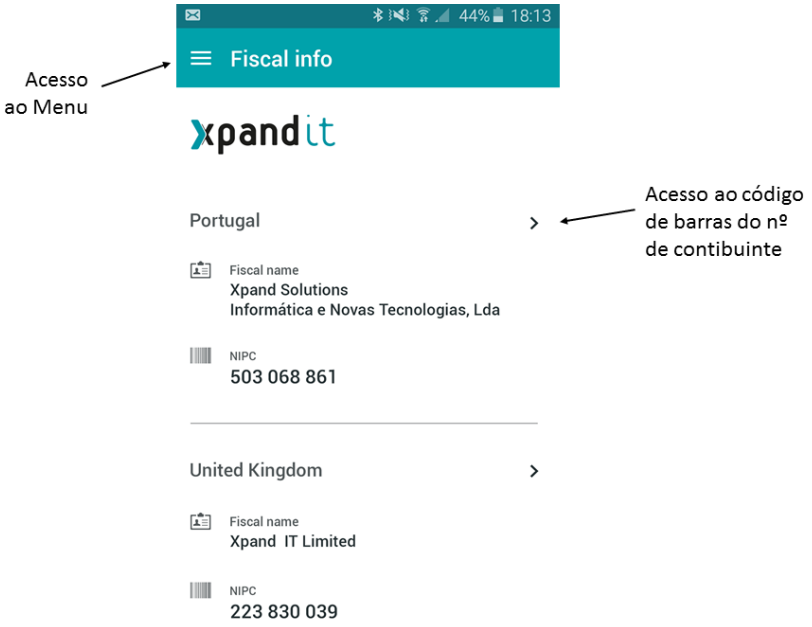

Figura 5.16: Xpack - Informação fiscal da Xpand IT

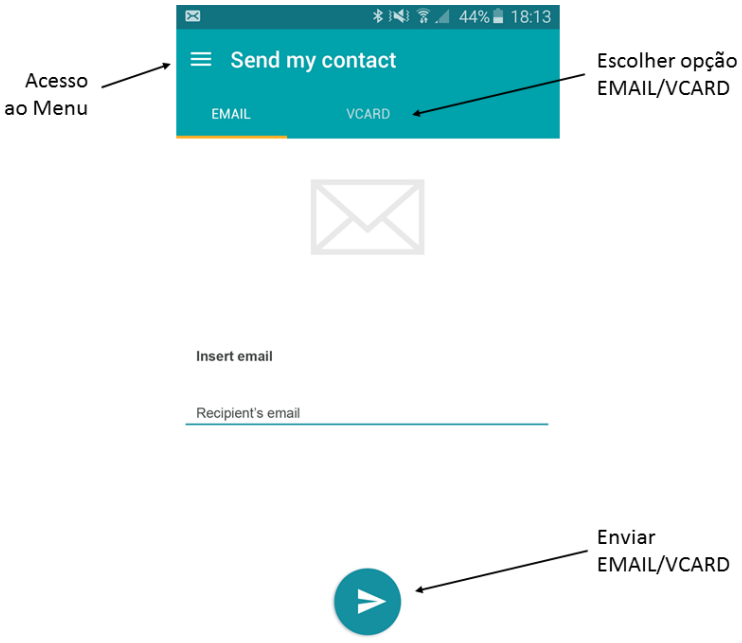

Figura 5.17: Xpack - Enviar contacto por email

5.3.4 Modelos de dados

Os modelos de dados usados neste projeto são básicos, pois o resto da informação foi passada diretamente pelos *resources* das *ViewModels* como é o caso do ecrã de Fiscal Information. Assim sendo, foi usado um modelo para as informações dos escritórios Company – outro com toda a informação dos colaboradores Employee, e outro com a informação de cada uma das notícias FeedEntry. Foi criado também um Feed com a lista das notícias. Todos os atributos são públicos, de forma a poderem ser acedidos pelas *ViewModels*.

Entidade	Atributos
Company	+Id: String +Location: String +Address: String +Phone: String +Fax: String +Latitude: String +Longitude: String
Employee	+Id: String +FirstName: String +LastName: String +Name: String +DisplayName: String +Email: String +BirthDate: String +MobileNumbers: String +Hangout: String +Skype: String +Outlook: String +ImageUrl: String +Thumbnail: String +BusinessUnit: String
Feed	+Id: String +Title: String +Subtitle: String +Entries: ObservableCollection<FeedEntry>
FeedEntry	+Id: String +Title: String +AuthorName: String +Updated: String +Published: String +ImageUrl: String +Link: String +Text: String

Entidade	Atributos
TimesheetWeek	+Submitted: Boolean +Approved: Boolean +LastTimeEdited: DateTime +Days: List<TimesheetDay>
TimesheetDay	+Day: DateTime +TotalTimeEntries: String +Entries: List<TimesheetEntry>
TimesheetEntry	+TimeEntryId: String +ProjectName: String +TaskName: String +TimeEntryDate: DateTime +TotalTime: DateTime +Approved: Boolen +Submitted: Boolean +CostCenter: String +WorkType: String +Description: String
AbsenceDay	+Date: DateTime +DayType: Enum
AbsenceType	+Name: String +Color: String +Id: Guid +DaysAvailable: double

Tabela 5.10: Modelo de dados

Na segunda parte da tabela (parte direita), estão os modelos base usados pe-

las plataformas para as funcionalidades Timesheets e Ausências. É de notar, no entanto, que existem mais modelos que são usados como apoio a determinadas operações (por exemplo, para ir buscar os dados aos serviços). Porém, a tabela iria tornar-se demasiado extensa sem adicionar informação útil.

5.4 Detalhes de funcionalidades

Nesta secção irão ser descritos detalhes das funcionalidades Timesheets, Ausências e Notificações Push.

5.4.1 Sistema de *tracking*

Como foi referido anteriormente, foi construido um sistema onde é possível fazer o registo de todas as tarefas e projetos onde cada um dos colaboradores esteve envolvido (ver Fig. 5.18).

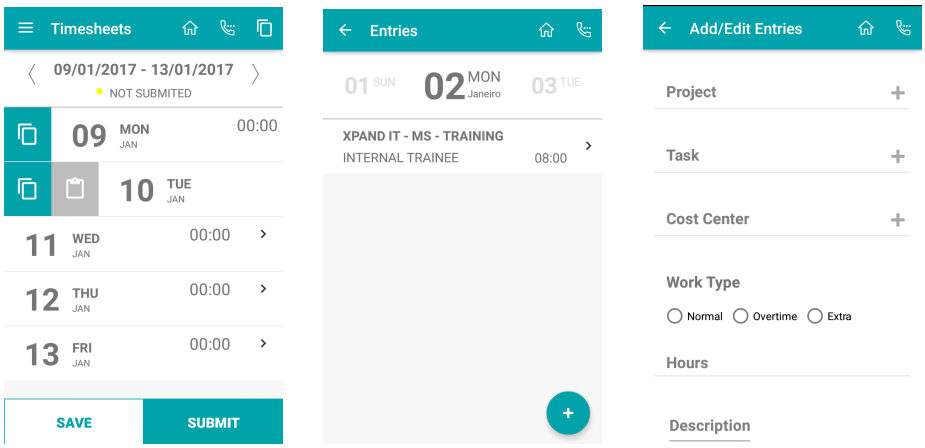

Figura 5.18: Ecrãs da funcionalidade Timesheets

O primeiro ecrã desta figura foi o ecrã que levou mais tempo a ser implementado a nível de *design* e funcionalidade, visto que não existia nenhum componente que permitisse a realização do "*double swipe right*"para copiar ou colar um dia da timesheet semanal da forma que estava planeado. Como tal teve de ser implementado tudo manualmente tanto na plataforma Android, como na plataforma

Windows. A função de copiar/colar os dados em concreto foi construida no "*Clipboard Service*" (ver secção 5.2.2).

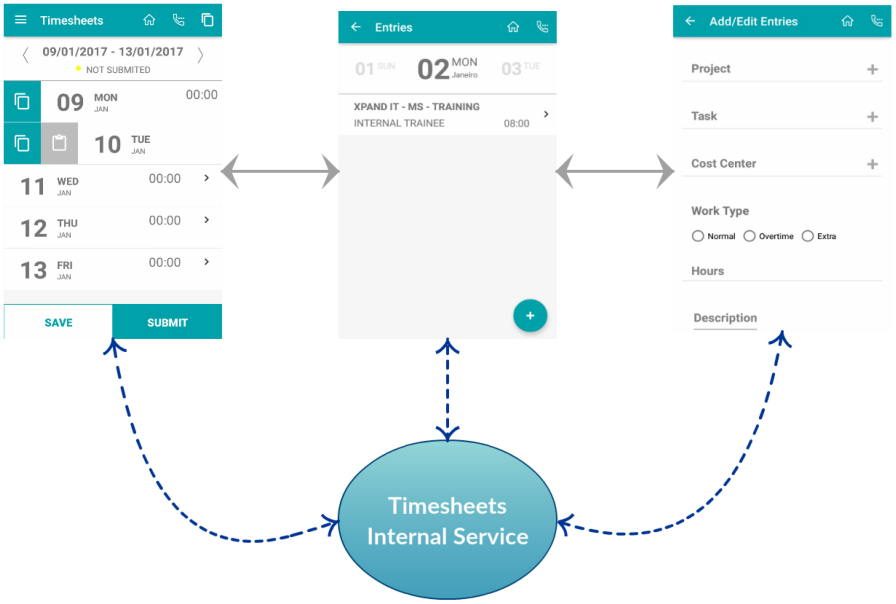

Figura 5.19: Serviço interno das Timesheets

Conforme referido na secção 5.2.2, foi desenvolvido um serviço interno no Core. Sempre que existe alguma alteração em qualquer sequência dos ecrãs referidos, tudo é gravado neste serviço. Quando é carregado um novo ecrã das timesheets, a informação é toda carregada através deste serviço e posteriores modificações são refletidas aqui. Para além disso, operações de copiar ou colar são também realizadas através deste serviço, para que todas as tarefas sejam realizadas de forma igual nas duas plataformas.

5.4.2 Sistema de ausências

Foi desenvolvido um sistema de ausências para que cada colaborador possa registar as suas ausências de forma fácil e eficaz. Existem dois ecrãs para essa funcionalidade:

- **Ecrã visão anual:** Neste ecrã temos uma vista anual para que possamos escolher o mês à qual pretendemos adicionar uma nova ausência. Podemos também, ao virar o ecrã para uma posição horizontal, ver um gráfico de

barras anual onde conseguimos comparar as ausências que foram marcadas até ao momento. As ausências que forem marcadas como férias, são visíveis neste ecrã através de um ícone no mês respetivo;

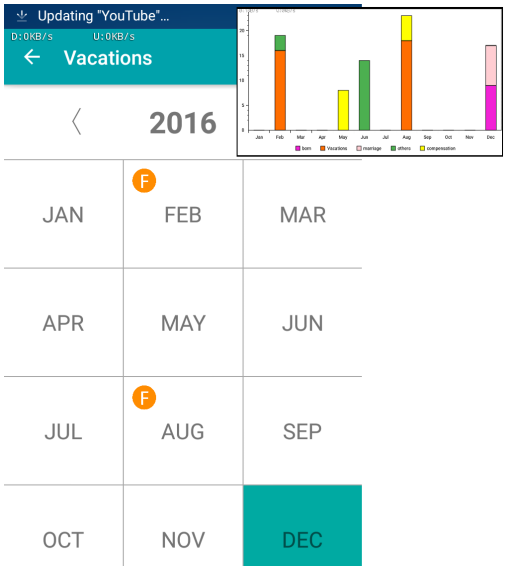

Figura 5.20: Ecrã de ausências: Visão Anual

- **Ecrã visão mensal:** Neste ecrã que temos uma visão mensal do calendário de ausências e onde conseguimos ver as ausências em detalhe que estão marcadas até ao momento. Para além disso, é possível marcar novas ausências e verificar o número de ausências que ainda é possível gastar por cada tipo.

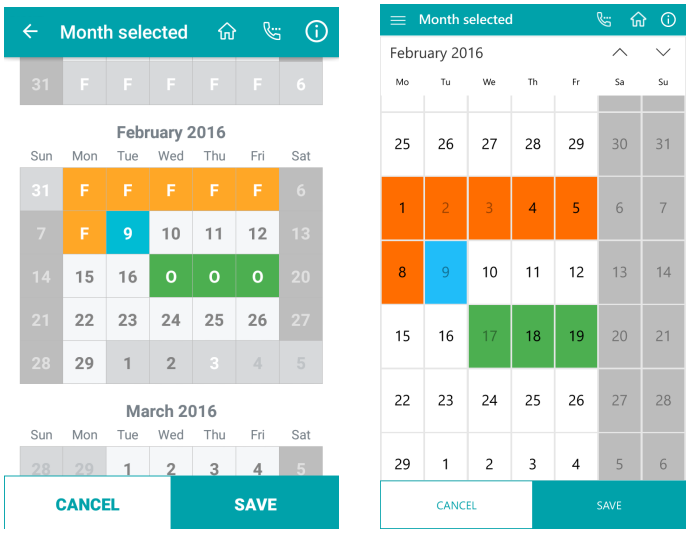

Figura 5.21: Ecrã de ausências: Visão Mensal

Na realização deste último ecrã (Fig. B.20, Android à esquerda e UWP à direita), houveram algumas diferenças nas duas plataformas a nível visual (relativas ao calendário). Isto porque no android foi utilizado um componente já implementado (Timesquare[3]) e no Windows foi usado o calendário nativo. Ao serem utilizados dois componentes tão distintos, verificou-se uma ligeira diferença a nível visual nas duas plataformas, no entanto, a interação é feita exatamente da mesma forma.

É de notar que na implementação do calendário do android, a biblioteca utilizada (Timesquare) é livre, o que permitiu que esta fosse bastante alterada a nível de código para conseguir o aspeto e a interação pretendida. O mesmo aconteceu no calendário nativo do Windows que sofreu diversas alterações para conseguir o aspeto final (apesar de ser mais limitativo, visto que não permitiu, por exemplo, a alteração das letras dos dias).

5.4.2.1 Ausências *Web Service*

Na Xpand IT existe o serviço Timesheets onde é feito todo o *tracking* das horas semanais e projetos em que cada colaborador esteve incluído. Esse serviço

[3]Ver: `https://components.xamarin.com/view/square.androidtimessquare` (acedido a Janeiro de 2017)

está construido num *software* já existente – TimeLive[4]. Este *software* já incluía a componente do registo, consulta e aprovação de qualquer tipo de ausências. No entanto não estava a ser usado pela Xpand IT visto que eram necessárias alterações a nível do cálculo anual da contabilização de dias de férias. Para isso a Xpand IT comprou o código à LiveTecs (para poder ser editado) e foi então adicionada essa componente para que a contabilização das ausências no TimeLive funcionasse de acordo com as normas da empresa. Assim, e como fazia sentido, o serviço das férias foi construido dentro deste serviço. Tudo isto para explicar que não foi construido um serviço de base, mas foi integrado num serviço já existente.

Assim, foi construida apenas mais um *web service* em Visual Basic (linguagem à qual este sistema se encontrava desenvolvido) com três métodos:

- **Get Absence Types:** [GET] recebe o *mail* do colaborador e vai procurar os tipos de ausência atribuídos a este utilizador, assim como o número restante de ausências que tem disponível para cada tipo; devolve o resultado em JSON com os dados correspondentes;
- **Get Absence List:** [GET] recebe o *mail* do colaborador e vai procurar a lista das ausências já submetidas pelo utilizador; devolve as ausências marcadas numa lista em JSON com os dados correspondentes;
- **Submit New Dates:** [POST] recebe o *mail* do colaborador e a lista das novas ausências a serem submetidas; após isso os dados são adicionados à base de dados e é devolvido um *"HTTP Status OK"* a informar que os dados foram adicionados com sucesso.

Na figura 5.22 podemos ver como o serviço está a funcionar em detalhe. Quando, por exemplo, o serviço recebe um pedido GET dos tipos de ausência, o serviço vai procurar estas na base de dados Microsoft SQL através de *scripts* T-SQL[5]. Assim sendo, não é escrito qualquer tipo de código para fazer operações sobre a base de dados visto que já existem métodos pré-definidos que fazem as operações que são pretendidas para o objetivo deste serviço. É de salientar portanto que todo este tipo de *scripts* T-SQL já se encontravam implementados, não tendo sido produzido nenhuma linha de código para o acesso à base de dados.

[4] *Software* que permite fazer o *tracking* das horas (não sendo propriedade da Xpand IT, mas sim da LiveTects: `http://www.livetecs.com/`)

[5] Transact-SQL: permite a criação de *scripts* pré-definidos que fazem operações sobre a base de dados.

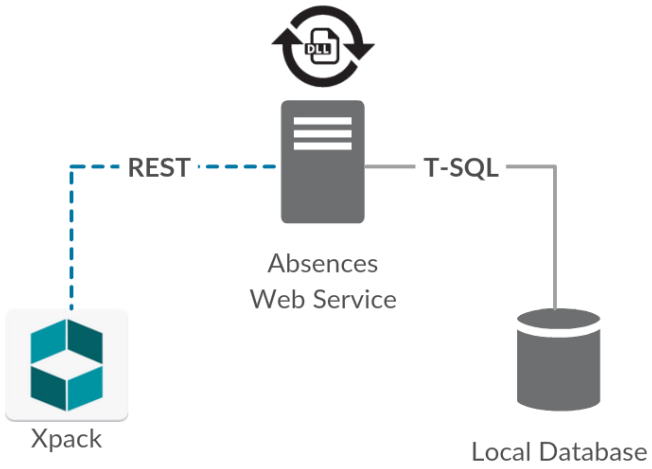

Figura 5.22: Absence Web Service

Como podemos ver nesta figura, existe um símbolo a cima do "*Absences Web Server*". Este símbolo representa a conversão de um modelo de dados. Isto porque o modelo de dados utilizado pelo TimeLive é diferente do modelo de dados usado na aplicação Xpack. Como tal foi criado um DLL (*Dynamic-Link Library*) que permite a conversão dos dados fornecidos pela base de dados do TimeLive para os dados a serem consumidos pela Xpack (e vice versa).

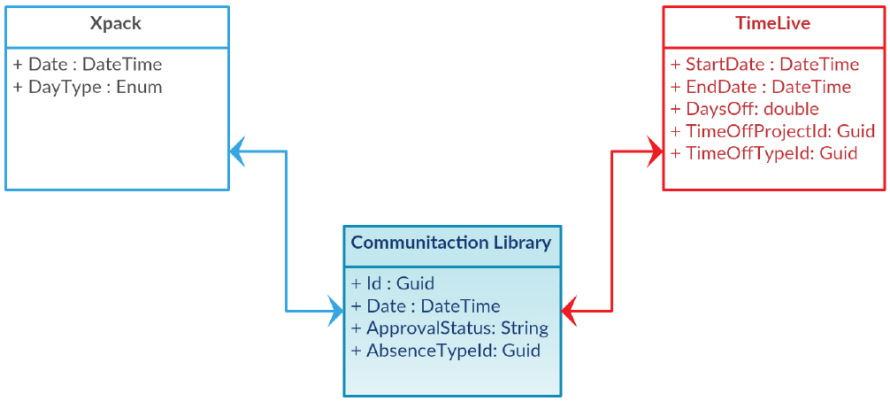

Figura 5.23: Communication Library

A figura 5.23 retrata o modelo de dados usado na Xpack e no TimeLive. Assim, sempre que a Xpack precisa de comunicar com o TimeLive, os dados são primeiro convertidos para a "*Communication Library*"para posteriormente poder contactar o TimeLive (o contrário também acontece).

5.4.3 *Push Notifications*

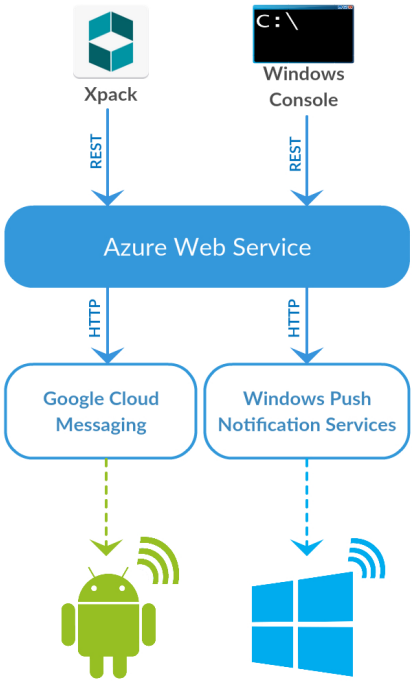

Figura 5.24: *Notification Service*: Diagrama Geral

As *Push-Notifications* são um sistema de notificações que se baseiam num tipo de tecnologia diferente, em que a notificação em vez de ser estimulada por uma razão interior, é estimulada através de um servidor exterior (normalmente através da Internet). Deste modo, é possível enviar uma notificação para o utilizador, em tempo real, as vezes que forem necessárias. O utilizador irá receber a notificação, mesmo que a aplicação esteja fechada. Caso este não esteja ligado à Internet, a notificação irá ficar em lista de espera e, assim que, o utilizador se conectar à Internet recebe as notificações que estiverem pendentes.

Neste caso, vai ser possível enviar uma *Push Notification* através de duas formas distintas:

- **Via Xpack:** através do login em modo de administrador na aplicação Xpack, aparecerá uma opção onde será então possível enviar uma *push notification*;
- **Via backoffice:** através de uma aplicação Windows que tratará de enviar a *push notification*.

Podemos ver com base na figura 5.24, que é então possível enviar as notificações através da aplicação Xpack, ou de um terminal Windows. O que acontece em qualquer um dos casos, é que a aplicação comunica via REST com um serviço que foi criado no *Azure Web Service* que se limita a receber uma *string* e a comunicar com os serviços das plataformas. No caso do Google Cloud Messaging, trata-se de um serviço de notificações específico, desenvolvido pela Google e está responsável por notificar todos os dispositivos Android que tenham uma aplicação que esteja subscrita a um canal de notificações identificado por uma chave exclusiva.

De forma semelhante, o mesmo acontece em Windows com o *Windows Push Notification Services*. Neste caso, a aplicação é identificada por uma *Package SID* e uma *Security Key*, que serve como autenticação nos serviços de notificação Windows.

É ainda possível enviar uma mensagem específica para apenas determinados utilizadores através de uma *tag*. Ao enviar uma notificação, é possível colocar uma *tag* e só os utilizadores que estejam subscritos a essa determinada *tag* é que recebem a notificação. Isto é útil para diversas situações, nomeadamente para quando pretendemos notificar os colaboradores de apenas um departamento, ou de um determinado projeto. Este sistema está disponível não só na *Google Cloud Messaging*, como também no *Windows Push Notification Service*.

Na figura 5.25 temos um retrato do processo de notificação passo-a-passo. Aqui estão retratadas duas situações em simultâneo: primeiro o registo do dispositivo no respetivo serviço de notificações, e em segundo, o processo de envio de uma notificação. Na figura, é possível identificar os vários passos, onde os números repetidos representam o mesmo processo, quer em Android ou Windows.

1. – [Android] É enviado um pedido de registo no *Google Cloud Messaging*;
   – [Windows] É enviado um pedido de canal de notificações *Push* ao *Windows Push Notification Service*;

2. – [Android] O *Google Cloud Messaging* devolve um ID de registo e a ligação é estabelecida;
   – [Windows] O *Windows Push Notification Service* cria o canal e devolve este em forma de URI (*Uniform Resource Locator*);

3. – [Android e Windows] Envio de um pedido via REST ao *Web Service* com a mensagem a enviar na *Push Notification*;

4. – [Android] O *Web Service* envia a mensagem para o *Google Cloud Messaging* com a respetiva *API KEY* que está responsável por identificar a aplicação

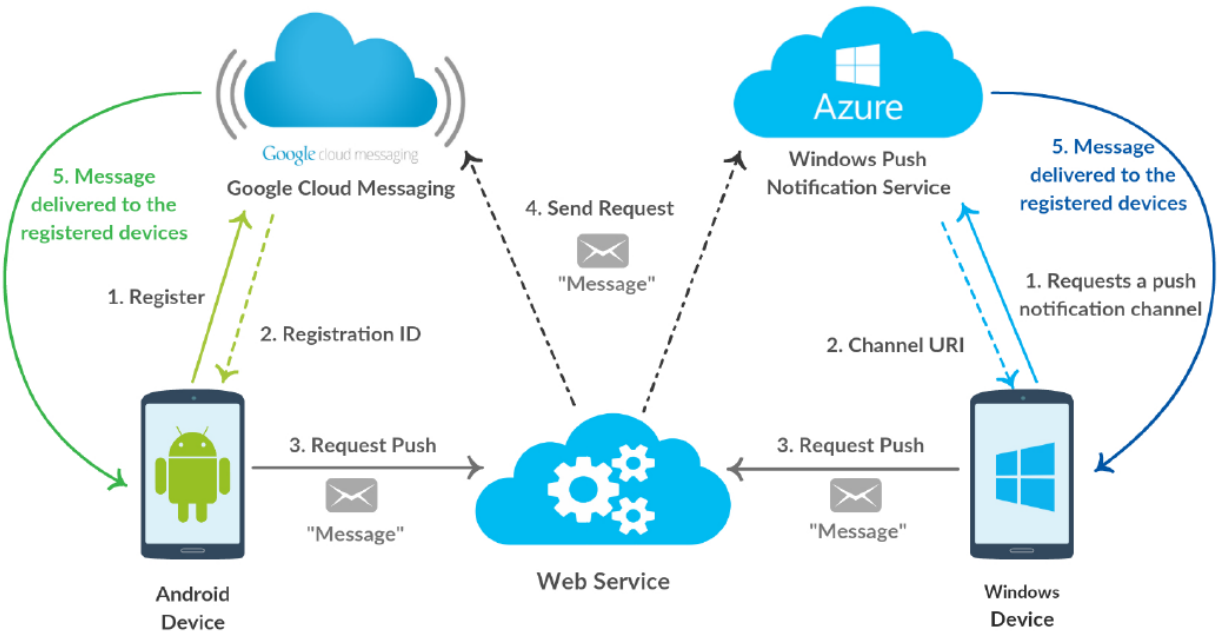

Figura 5.25: *Notification Service diagram*: passo-a-passo

à qual se prentende enviar notificação; [Windows] É enviada a mensagem também para o *Windows Push Notification Service* através do *channel URI*. Isto é feito através de um pedido HTTP POST, usando o SSL[6];

5. – [Android] O *Google Cloud Messaging* recebe e envia a respetiva mensagem para todos os dispositivos Android que estejam registados; [Windows] Também o *Windows Push Notification Service* recebe e envia a respetiva mensagem para todos os dispositivos Windows que estejam registados.

5.4.3.1 *Notification Web Service*

Como podemos ver na figura 5.24 e na figura 5.25, foi desenvolvido no escopo desta dissertação, um serviço externo exclusivo para o funcionamento deste sistema de notificações. Este serviço está alojado no Azure, para evitar preocupações a nível de *hardware* e *performance*. Como já conseguimos perceber pelas ilustrações anteriores, o que este serviço faz é receber pedidos para lançar notificações. Seja pela Xpack, seja pelo terminal Windows, este serviço recebe pedidos via REST e processa o pedido enviando a mensagem para os serviços *Google Cloud Messaging*

[6]Secure Sockets Layer

e *Windows Push Notification Service*, de forma a notificar os dispositivos Android e Windows, respetivamente.

5.5 Estrutura das plataformas

Esta secção serve para esclarecer como é que as plataformas Android e Windows estão organizadas internamente para apresentar os ecrãs de forma lógica.

5.5.1 Android

No Android a aplicação está estruturada com apenas duas Activities diferentes:

- **Login Activity:** responsável por tratar do ecrã de início de sessão e respetivas operações;
- **Home Activity:** responsável por todos os restantes ecrãs da aplicação; Isto significa que esta é sempre a *activity* base que é usada depois do início de sessão. Todos os restantes ecrãs estão agregados a esta *activity* através da criação de fragmentos.

A aplicação foi assim estruturada de modo a facilitar algumas tarefas: por exemplo, com os fragmentos é possível transmitir informação de ecrã para ecrã; isto facilita bastante a programação quando temos sequências de ecrãs que necessitam de informação do ecrã anterior. É de notar que cada fragmento ou activity (no caso do Login Activity) tem um *layout* que corresponde a um "axml"com a parte visual do ecrã. Assim sendo estas classes funcionam como *controller* dessa parte visual.

5.5.2 Windows

No Windows o esquema de apresentação de ecrãs funciona de uma forma ligeiramente diferente, sendo que cada ecrã corresponde a uma *view* que tem um *controller* e um ficheiro "axml"que corresponde à parte visual da aplicação. Assim sendo, cada ecrã diferente corresponde a uma *view* diferente, e ao contrário do Android com os fragmentos, não existe qualquer ligação entre as diversas *views*.

Capítulo 6

Testes

Assegurar a qualidade do *software* produzido é uma das componentes mais importantes no desenrolar de um projeto. Como tal existiu a necessidade de fazer diversos testes de modo a garantir esta componente, entre eles testes de usabilidade e testes de aceitação. Apesar da realização destes dois tipos de testes, houve também "*ad-hoc testing*" em que sempre que era desenvolvida uma funcionalidade, esta era testada pelo menos com os casos de testes mais básicos.

6.1 Testes de usabilidade

Os testes de usabilidade têm como objetivo colmatar algumas falhas especialmente a nível de *design*, apesar de por vezes se descobrir alguns problemas também a nível do *software* em si. Para além disso, este tipo de testes permite descobrir se o propósito da realização de uma dada funcionalidade foi de facto conveniente ou se não cumpriu os objetivos inicialmente propostos. Estes testes foram realizados no final do desenvolvimento de todas as funcionalidades, e com utilizadores semelhantes aos potenciais utilizadores finais, em que se pediu a cada um destes que fizesse uma determinada tarefa. Foi observado todo o comportamento do utilizador e apontado o número de cliques que este efetuou, assim como o tempo demorado a realizar a dada tarefa. A seguir vão ser apresentados os dados finais da experiência e respetivas conclusões. As tabelas e dados respetivos a cada uma das experiências podem ser consultadas no Anexo C.

É importante referir que foram selecionadas apenas algumas tarefas consideradas mais importantes, não abrangendo a totalidade de funcionalidades da Xpack,

com o intuito de não tornar o teste demasiado comprido. Foi pedido a cada um dos inquiridos que realizasse cada uma das seguintes tarefas:

- **T01:** Iniciar sessão na aplicação;
- **T02:** Aceder à morada do escritório de Londres;
- **T03:** Aceder à informação fiscal da empresa e obter o código de barras do número fiscal da Xpand Portugal;
- **T04:** Aceder aos contactos e procurar o contacto "Paulo Lopes". Adicionar este contacto à lista telefónica do telemóvel;
- **T05:** Aceder à funcionalidade de ausências e marcar uma ausência do tipo "férias"de 3 a 6 de Fevereiro de 2017;
- **T06:** Aceder à funcionalidade Timesheets e adicionar uma nova entrada no dia atual;
- **T07:** Ainda na funcionalidade anterior, copiar todas as entradas da semana passada para a semana atual;
- **T08:** Mudar o Cost Center da entrada que tinha sido adicionada na tarefa T06;
- **T09:** Aceder à informação de perfil do utilizador e mudar a palavra passe;
- **T10:** Enviar a minha informação de contacto para um dado *email*;
- **T11:** Terminar sessão na aplicação.

No total foram inquiridas seis pessoas, número este considerado suficiente dado que os padrões de cliques/tempo tomado e comentários feitos por estes, repetiu-se com bastante frequência.

Na Tabela 6.1 estão retratados o número de cliques estimados que era necessário para realizar a tarefa em questão *versus* o número de cliques, em média (arredondada para cima), efetuados pelos inquiridos. Na Tabela 6.2 está retratado o tempo que seria necessário para realizar a tarefa *versus* o tempo, em média, tomado pelos inquiridos. Apenas por estas duas tabelas conseguimos concluir por certo que existe uma tarefa que foi bastante mais complicada em relação às outras: estamos a falar da Tarefa T07, com um número de cliques estimados de 4, contra 23 cliques realizados; assim como apenas seria necessário 6 segundos para realizar esta

tarefa, quando em média o tempo decorrido foi de 2 minutos e 3 segundos. Para além disso, e como é possível ver nos anexos, esta foi a única tarefa que ninguém conseguiu concluir com sucesso, sendo que todos os inquiridos desistiram.

Nas restantes tarefas, notaram-se alguns ligeiros desvios mas algo considerado normal dado o facto que os inquiridos nunca tinham interagido com esta aplicação em concreto.

ID	Cliques estimados	Cliques realizados (média)
T01	3	4
T02	3	6
T03	3	3
T04	5	7
T05	8	13
T06	14	18
T07	4	23
T08	6	13
T09	6	14
T10	5	12
T11	2	2

Tabela 6.1: Cliques estimados *versus* cliques realizados (em média)

ID	Tempo estimado	Tempo demorado (média)
T01	00:15	00:44
T02	00:07	00:36
T03	00:04	00:09
T04	00:17	00:35
T05	00:12	01:05
T06	00:42	02:01
T07	00:06	02:03
T08	00:13	00:54
T09	00:18	01:32
T10	00:27	01:53
T11	00:02	00:07

Tabela 6.2: Tempo estimado *versus* tempo demorado (em média)

Com os testes de usabilidade concluídos, existem já várias conclusões e sugestões feitas pelos próprios inquiridos. Esta é a razão e o fator número um para a realização deste tipo de testes. Podemos então retirar as seguintes conclusões de cada uma das tarefas:

- **T04:** Nesta tarefa os inquiridos sugeriram que aparecesse um pop-up a avisar que o contacto teria sido adicionado. Caso contrário eles não iriam ter a certeza que a tarefa teria sido executada com sucesso (visto que maior parte deles foram à lista de contactos do telefone confirmar). Para além disso, na totalidade dos inquiridos, apenas um se apercebeu da existência de um botão de procura, ao invés dos restantes que foram manualmente à procura do contacto pedido. Quer isto dizer que a procura de contacto deve ficar mais óbvia de alguma forma;
- **T05:** Esta foi uma tarefa bastante sensível e com vários pontos a melhorar. Primeiro ponto: esta funcionalidade não está suficientemente visível visto

que todos os utilizadores fizeram diversos percursos diferentes na aplicação até chegarem ao ecrã "Apps"(ecrã onde se encontra o acesso à funcionalidade pretendida) e apenas chegaram a esse ecrã por exclusão de partes. Segundo ponto: a marcação de uma ausência não está intuitiva. Isto é, ao carregar num dia do calendário aparece um pequeno balão no canto inferior do ecrã a avisar para escolher o último dia do intervalo a marcar. No entanto, na grande maioria os inquiridos não se aperceberam desse facto, resultando na marcação de dias sem terem intenção de tal. Deve, portanto, haver uma alteração a este nível para tornar a marcação dos dias mais intuitiva;

- **T07:** Esta foi a tarefa mais complicada de todas as outras e que ninguém concluiu com sucesso. Para realizar esta tarefa há um botão no canto superior direito que o permite fazer com apenas dois cliques. Apesar de alguns utilizadores terem chegado a carregar nesse botão, não perceberam que seria ali que poderia copiar a semana inteira. Isto porque não existe nenhuma informação que o indique. Assim, existem dois pontos a melhorar: acesso a esse botão; e tornar o botão intuitivo o suficiente ou com informação suficiente para os utilizadores perceberem que esse botão serve para realizar a tarefa pretendida;

- **T09:** Relativamente a esta operação, os inquiridos na sua grande maioria, não perceberam à primeira vista que seria possível realizar essa tarefa dentro do ecrã "Editar Perfil". Daí terem levado mais tempo e feito mais cliques do que era suposto, apesar de todos terem conseguido concluir a tarefa com sucesso. Foi sugerido que houvesse uma secção "Settings"onde se pudesse fazer este tipo de operações;

- **T10:** Esta foi uma tarefa fácil de realizar de um modo geral, tendo havido apenas uma sugestão: poderia haver algum botão na lista de contactos (em cada um dos contactos) em que fosse possível enviar aquele contacto para alguém, evitando assim a existência de um ecrã exclusivo para esta operação, e acrescentando o facto de se poder enviar não só o nosso contacto, como também outros contactos.

No final do teste de cada inquirido, foi ainda pedido que avaliasse cada uma das seguintes frases de acordo com o que pensava de 1 a 10, sendo que 1 representava "Discordo totalmente"e 10 representava "Totalmente de acordo"(ver Tabela 6.3). Com estes resultados podemos concluir, que apesar de terem havido algumas dificuldades em determinadas tarefas, o *feedback* final é positivo e a experiência é bastante satisfatória.

	I01	I02	I03	I04	I05	I06	Média
"A aplicação tem uma interface amigável"	5	9	10	8	8	8	**8**
"É fácil fazer as tarefas mais básicas e importantes"	6	10	10	7	8	8	**8**
"De certeza que iria utilizar esta aplicação para realizar as operações que esta me permite"	8	10	10	8	10	9	**9**

Tabela 6.3: Testes de usabilidade: avaliação da Xpack

6.2 Testes de aceitação

Os testes de aceitação são testes que têm como função principal testar a aplicação como se esta fosse uma caixa negra. Isto é, é conhecido desde início que vão existir determinadas tarefas que são executadas na aplicação e no final da realização de cada tarefa é esperado um resultado.

Para a realização deste tipo de testes, foi recorrido à ferramenta que foi falada no início desta dissertação (ver capítulo 1) – Xamarin Test Cloud[1] – que providencia uma forma de realizar os testes de aceitação em milhares de dispositivos moveis reais. Tem no entanto como desvantagem, o facto de só estar disponível para dispositivos Android e iOS, razão pela qual apenas a aplicação Android ter sido alvo destes testes.

Na grande maioria das vezes que os programadores estão a desenvolver aplicações para dispositivos móveis, é comum testar a aplicação num número de dispositivos limitados (normalmente inferior a três). O objetivo desta alternativa é testar a aplicação num grupo mais alargados de dispositivos. Isto porque o facto de uma aplicação funcionar num dispositivo, não implica necessariamente que possa funcionar noutro dispositivo diferente de forma igual. Assim, é possível com mais certeza afirmar que a aplicação funciona na grande maioria dos dispositivos móveis alvo.

Ao realizar este tipo de testes no Xamarin Test Cloud, é possível escolher o número de dispositivos móveis que desejarmos com as configurações que pretendermos, seja:

- Versão do sistema operativo;
- Características físicas do dispositivo (velocidade do processador, número de

[1] `https://www.xamarin.com/test-cloud`

cores do processador e memória);

- Marca do dispositivo;
- Modo tablet ou modo telemóvel.

É de salientar que esta é uma ferramenta é paga, e para fazer estes testes foi usada uma conta do tipo *trial* que apenas permite fazer os testes em três dispositivos móveis diferentes. Assim, foram alvo de testes os seguintes dispositivos:

- LG G3 s (Android 4.4.2);
- Motorola Nexus 6 (Android 5.1);
- HTC One (M8) (Android 4.4.3).

Na Tabela 6.4 está descrita a descrição de cada um dos testes com o comportamento que é esperado que aconteça com a realização de cada um dos testes de aceitação. É de notar, que para estes testes não foi incluída a funcionalidade de ausências pelo facto de esta necessitar de uma ligação VPN (*Virtual Private Network*) para poder funcionar (não sendo possível com o Xamarin Test Cloud).

Tabela 6.4: Testes de aceitação

ID	Descrição do teste	Comportamento esperado
AT01	**Início de sessão** com o utilizador errado / não preenchido.	A sessão não é iniciada e aparece uma mensagem popup a informar que as credenciais estão incorretas.
AT02	**Início de sessão** com a password errada / não preenchida.	A sessão não é iniciada e aparece uma mensagem popup a informar que as credenciais estão incorretas.
AT03	**Início de sessão:** O primeiro teste consiste na introdução das credenciais de utilizador.	A sessão é iniciada e são apresentados os elementos do ecrã principal.
AT04	**Página principal:** É aberta uma noticia.	Uma webview é carregada com as informações da notícia.

Continua na próxima página

Tabela 6.4 – continuação da página anterior

ID	Descrição do teste	Comportamento esperado
AT05	**Informação Fiscal:** Através do menu, é aberto o ecrã de informação fiscal e é selecionado o código de barras do número de contribuinte em Portugal.	Aparece a informação fiscal da Xpand IT em Portugal e Inglaterra. De seguida é apresentado o código de barras do contribuinte de Portugal.
AT06	**Escritórios:** Através do menu é aberto o ecrã "*Offices*".	É apresentada a informação dos três escritórios da Xpand IT.
AT07	**Enviar informação de contacto:** Através do menu é aberto o ecrã "*Send My Contact*". Nesse ecrã é preenchido o campo de email e é enviado o contacto por Vcard. De seguida é carregado na tab "email"e é enviado por email.	Ao tentar enviar um Vcard deve surgir um aviso a alertar que o utilizador atual não tem permissões para enviar um Vcard (isto porque o teste está a ser feito com um utilizador sem autorizações para tal procedimento). Após tentar enviar email, é reencaminhado para o mail do dispositivo.
AT08	**Lista de contactos:** Através do menu é aberto o ecrã "*Contacts*". Após o ecrã carregar é selecionado um contacto da lista.	Após a lista de contactos ser carregada e o contacto ser selecionado, aparece um ecrã com as informações detalhadas desse contacto (entre elas número de telefone, skype, Linkedin, mail, etc.).
AT09	**Lista de contactos:** Em continuação do teste anterior, é feita uma pesquisa de um determinado contacto por texto. Após isso, o contacto é selecionado.	Depois de concluir a pesquisa aparece o contacto pesquisado (caso este exista) e é aberto um ecrã com as informações deste.

Continua na próxima página

Tabela 6.4 – continuação da página anterior

ID	Descrição do teste	Comportamento esperado
AT10	**Timesheets:** Na página inicial é selecionado o botão de "Timesheets". Após isso é adicionada uma nova entrada à timesheet da semana atual.	Depois de adicionar a nova entrada, é suposto que esta apareça no ecrã principal das timesheets e no detalhe desse dia.
AT11	**Timesheets:** Em continuação do teste anterior, é agora copiada a timesheet da semana anterior para a semana atual.	O conteúdo da timesheet atual é igual ao conteúdo da semana passada.
AT12	**Timesheets:** Em continuação do teste anterior, agora é guardada a semana atual.	O estado da timesheet está agora guardado e caso a app seja reiniciada os dados dessa semana continuam guardados.
AT14	**Perfil:** É aberto o ecrã de "Editar perfil"e são verificados todos os elementos do perfil. Após isso é adicionado um novo número de telefone.	As informações do perfil devem constar no ecrã e ao adicionar um número de telefone, este deve ficar visível no perfil.
AT15	**Perfil e Logout:** Em continuação do teste anterior é editada a password e de seguida é terminada a sessão para validar a nova password (fazer login novamente).	Ao mudar a password deve aparecer um alerta a dizer que a password foi mudada com sucesso. Após o alerta aparecer, é feito o logout. A sessão deve ser agora iniciada com as novas credenciais e apresentada a página principal.

Na Figura 6.1 está um *print* do resumo do teste efetuado à Xpack a partir do Xamarin Test Cloud. Podemos então verificar todas as condições do teste, assim como algumas características como o tamanho da aplicação (69.79 MB) e o pico de memória que esta atingiu durante os testes (407.15 MB). É de notar que este pico de memória é calculado em conjunto com as aplicações e serviços base do sistema operativo Android a funcionar – portanto este pico abrange memória base do sistema operativo + memória das aplicações e serviços base do android + aplicação Xpack. Podemos verificar que todos os testes passaram com sucesso, exceto um deles que ocorreu apenas no dispositivo da marca LG (Android 4.4.2).

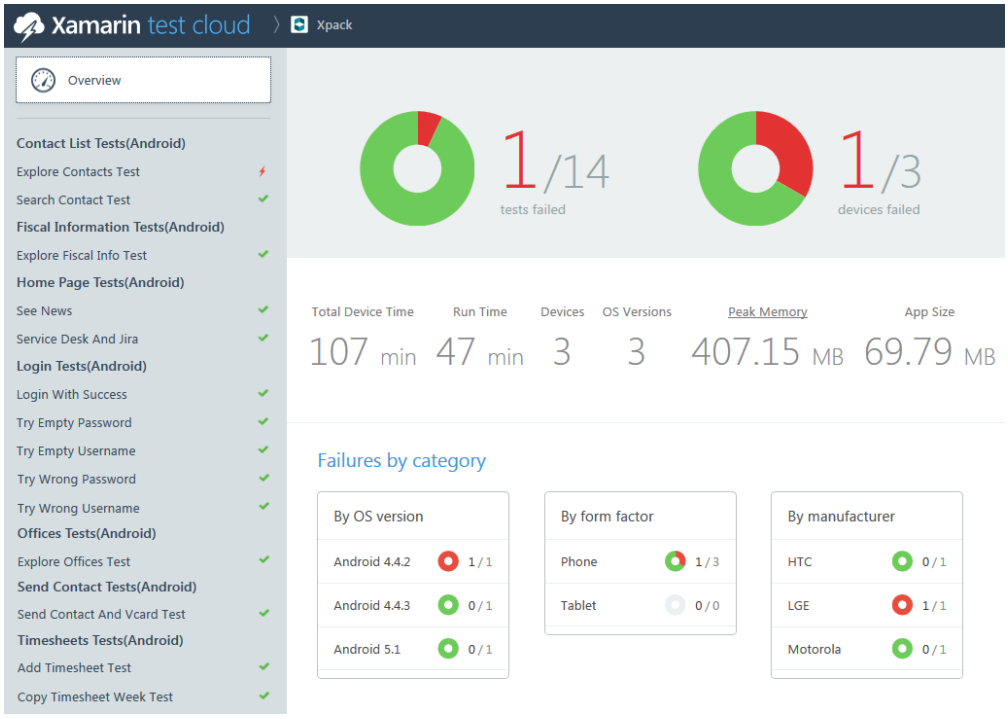

Figura 6.1: Xamarin Test Cloud – Resumo dos resultados

Na Figura 6.2 está retratada a falha que aconteceu no dispositivo Android. Esta falha ocorreu no teste AT08 (ver Tabela 6.4), quando a aplicação estava ainda no ecrã de início de sessão. A xamarin Test Cloud permite explorar melhor a falha e dá-nos a informação da "Exceção"que ocorreu – `"System.Exception : Error while performing EnterText(Id(‘‘userName"), "nrbm"); System.TimeoutException : Timed out waiting for keyboard to be shown."`. Quer isto dizer que ao carregar no campo "Username", o teclado para escrever não chegou a aparecer por alguma razão. No entanto, em todos os restantes testes correu tudo bem, o que quer dizer que pode ter ocorrido alguma falha no dispositivo que originou tal situação (é de notar que em todos os testes a aplicação é novamente instalada e iniciada como se fosse a primeira vez, querendo isto dizer que é sempre feito o início de sessão no início de cada teste).

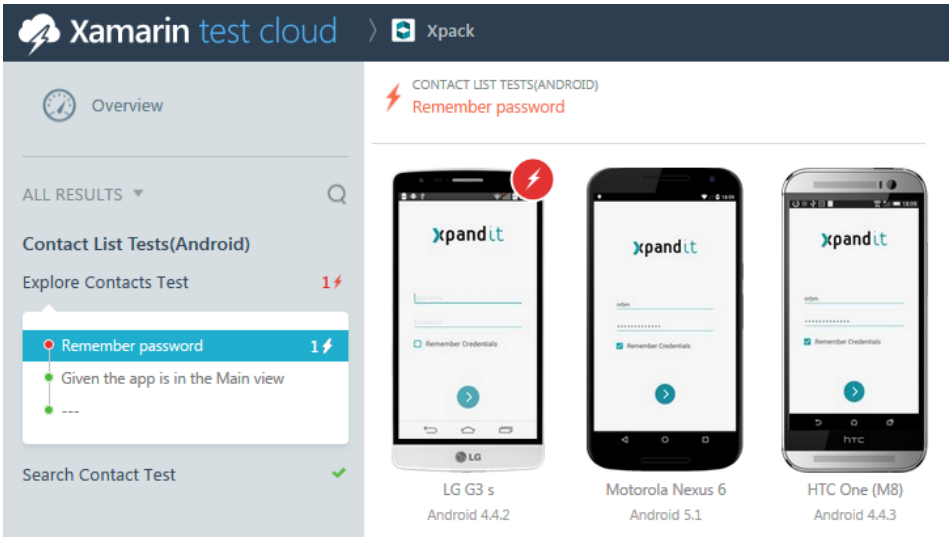

Figura 6.2: Contacts List Test

Na Figura 6.3 está retratado mais um exemplo de teste (neste caso teste à funcionalidade Timesheets) onde podemos verificar que é possível validar o ecrã dos dispositivos em cada uma das fases do teste, o que facilita bastante o trabalho do programador que consegue ver a sua aplicação em diversos dispositivos diferentes.

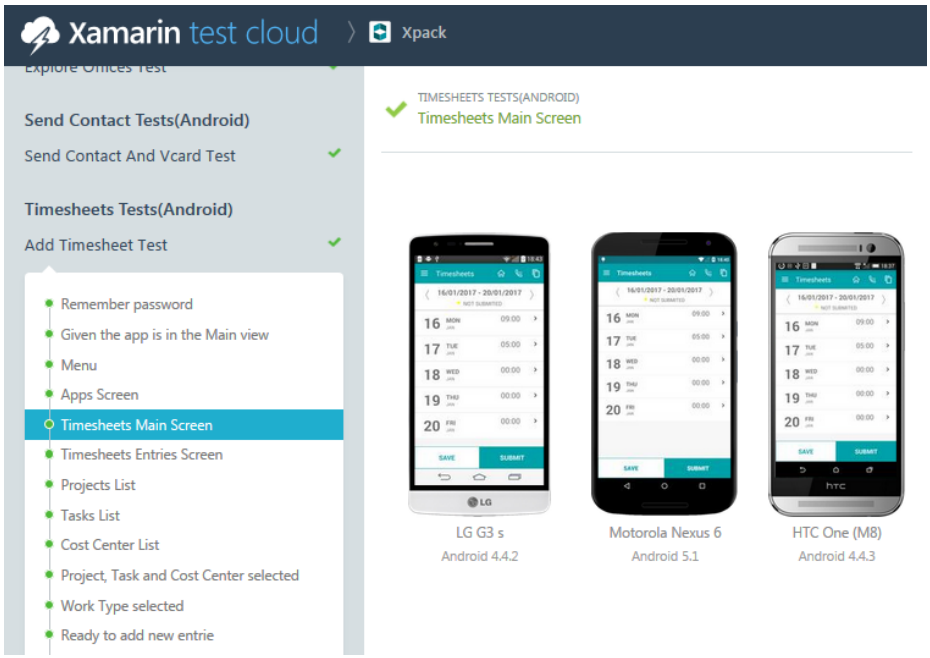

Figura 6.3: Timesheets Test

Concluindo, podemos validar que a aplicação comportou-se como era esperado em todas as situações a que esta foi submetida (tirando a falha no dispositivo LG), comprovando que de uma forma geral a aplicação está a funcionar de forma correta.

Capítulo 7

Resultados e conclusões

A primeira etapa do estágio foi constituída por fases de formação nas tecnologias a utilizar assim como investigação de outras tecnologias equiparadas a abordagens de desenvolvimento multi-plataforma para melhor entender as vantagens e desvantagens da abordagem que iria seguir. Após esta fase houve um levantamento dos requisitos para a aplicação móvel, e depois finalmente o desenvolvimento de uma primeira *release* da app. De uma forma geral, dado o panorama inicial que estava planeado para o desenrolar do semestre tudo correu conforme planeado.

O desenvolvimento desta *release* utilizando Xamarin, proporcionou uma oportunidade para constatar diversas vantagens desta alternativa, nomeadamente o facto de facilitar bastante o desenvolvimento de quem pretende aplicações destinadas a diversas plataformas, sem com isto comprometer a qualidade das aplicações desenvolvidas. Para além disso, o desenvolvimento em contexto empresarial foi enriquecedor a nível pessoal, nomeadamente no que toca à obtenção de experiência com a aplicação de metodologias *agile* praticadas na empresa.

Para a segunda parte do estágio estava planeada uma pequena fase de formação em testes, seguida da construção de *scripts* de teste para a primeira *release* da aplicação. Após esta fase, foi feito um levantamento de requisitos para as novas funcionalidades a implementar e o desenvolvimento destas. Houveram, no entanto, algumas mudanças que ocorreram no decorrer desta segunda fase do estágio por diversos motivos:

- A funcionalidade Timesheets demorou mais tempo do que o que foi estimado no cronograma, o que se verificou num atraso da implementação das funcionalidades seguintes;

- Como agravante, estava planeado da parte da empresa que o serviço das ausências, que comunica com a Xpack, estivesse implementado até à data do desenvolvimento desta funcionalidade; Porém, tal não aconteceu pela indisponibilidade da equipa que gere este sistema, dado o destino do produto e as prioridades do mesmo; Assim sendo este serviço foi acrescentado à lista de tarefas a desenvolver no escopo desta dissertação;

- Por último, houve a necessidade do desenvolvimento de uma funcionalidade extra (a pedido da equipa dos recursos humanos da empresa): sistema de notificações push, o que obrigou a estudar um novo tema para desenvolver esta funcionalidade e, também ao desenvolvimento do serviço de notificações que comunica com a Xpack.

Assim, por todas estas razões, o cronograma não correspondeu exatamente à realidade apesar de ter sido feito um esforço para o cumprir. Na Figura 7.1 está o resultado do segundo semestre, num diagrama Gantt. Como podemos ver as principais diferenças para a previsão feita em Setembro de 2016 são:

- O desenvolvimento da escrita da dissertação passou a ocupar o semestre inteiro até aproximadamente dia 21 de Janeiro, visto que sempre que foi desenvolvida uma funcionalidade, esta foi de imediato descrita no documento;

- A funcionalidade Timesheets passou a ocupar mais tempo, até aproximadamente 15 de Novembro;

- A funcionalidade de Férias/Ausências começou mais tarde e acabou mais tarde, até aproximadamente 17 de Dezembro;

- Foi acrescentado o desenvolvimento do serviço das ausências que começou aproximadamente a 19 de Novembro e acabou a 3 de Dezembro. Como podemos verificar esta tarefa está a ser executada ao mesmo tempo da funcionalidade de férias. Isto deve-se ao facto de durante o desenvolvimento, ter havido a necessidade de desenvolver o serviço para conseguir trabalhar os dados;

- A funcionalidade de editar perfil demorou ligeiramente menos tempo do que o esperado, tendo assim acabado ligeiramente mais cedo;

- De seguida foram acrescentadas duas novas tarefas que foram desenvolvidas em simultâneo visto que uma está dependente da outra: Notificações push e serviço de notificações push;

- A melhoria da arquitetura foi reduzida por falta de tempo, no entanto foram feitas as alterações essenciais para o projeto ficar mais organizado, sendo que no final o resultado é muito satisfatório, comparando inclusivamente com outros projetos da empresa;

- Por fim veio o desenvolvimento dos *scripts* de automação para o Test Cloud que foi desenvolvido a meados de Janeiro.

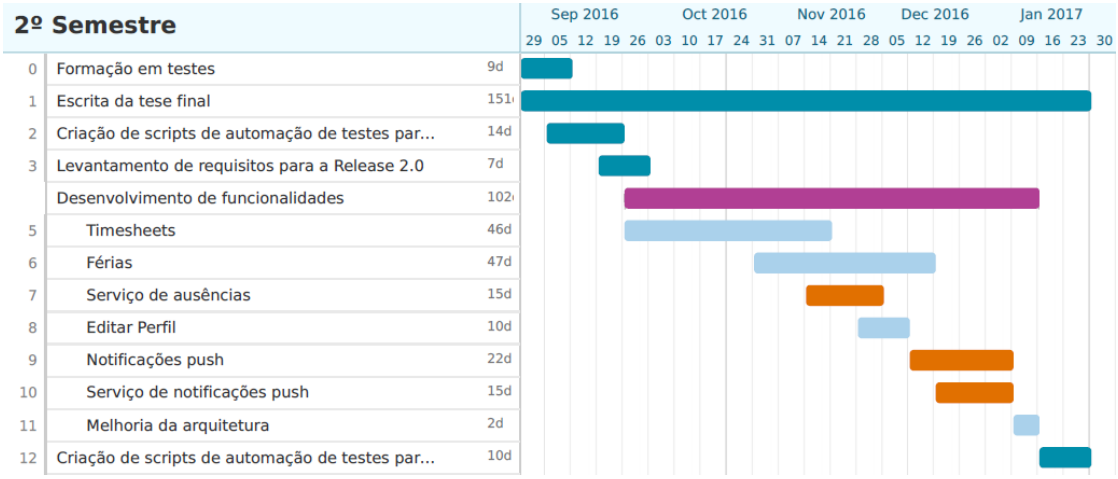

Figura 7.1: Gantt respetivo à realidade do segundo semestre

É conclusivo que na previsão do Gantt que foi feito para o segundo semestre, em Setembro de 2016, apenas contemplava o desenvolvimento de funcionalidades até ao final de Dezembro. No entanto com as razões acima descritas, as tarefas passaram a ocupar também Janeiro, até ao momento de entrega da dissertação. Na Tabela 7.1, está retratada a previsão e a realidade do tempo que cada uma das tarefas demorou a desenvolver, sendo que a maior mudança que se notou foi de facto o desenvolvimento das funcionalidades que passou de 80 para 102 dias, e a escrita da dissertação que passou de 122 para 151 dias.

Tarefa	Previsão	Realidade
Formação em testes	8 dias	9 dias
Criação de scripts para a R1.0	13 dias	14 dias
Levantamento de requisitos para a R2.0	6 dias	7 dias
Desenvolvimento de funcionalidades	80 dias	102 dias
Criação de scripts para a R2.0	6 dias	10 dias
Escrita da tese final	122 dias	151 dias

Tabela 7.1: Previsão do cronograma *versus* realidade

7.1 Requisitos

No Capítulo 4, foram descritos todos os requisitos funcionais e não funcionais para o desenvolvimento desta aplicação. Esta secção explica como é que cada um dos requisitos não funcionais (secção 4.2) foram ou não cumpridos:

- **NF01** (Interface): A interface foi estudada em conjunto com a equipa de *design*, que permitiu criar ecrãs fáceis de compreender e apelativos. No entanto, como pudemos concluir no Capítulo 6, de acordo com os testes de usabilidade, sentiu-se por vezes alguma dificuldade em realizar algumas tarefas que não eram óbvias o suficiente aos olhos dos utilizadores que foram alvos do teste;
- **NF02** (Segurança e privacidade): Para a validação deste requisito, foi implementada a encriptação das palavra passe do utilizador que está a usar a Xpack, a fim de esta deixar de estar guardada em *plain text*. Para além disso, todos os serviços que comunicam com a Xpack requerem autenticação, o que permite que para além de não haver fuga de dados, também está assegurado que estes não possam ser alterados;
- **NF03** (Lógica de negócio do código): Este requisito é considerado crucial no desenvolvimento de uma aplicação em Xamarin, visto que potencia uma das maiores vantagens que este tipo de abordagem permite – partilha de código, que permite poupança de código e por sua vez poupança no tempo de desenvolvimento – e permite obter uma hierarquia organizacional do código mais consistente. Por esta razão, este requisito esteve sempre presente no desenvolvimento de todas as funcionalidades desta aplicação e foi escrita, portanto, a maior quantidade de código quanto possível no projeto Core, a fim de permitir que este requisito fosse cumprido;

- **NF04** (*Performance* e Responsividade): Este requisito está de alguma forma ligado ao requisito anterior, visto que se a lógica de negócio estiver a ser cumprida, irá haver de imediato uma melhora na performance visto que não existe código a ser duplicado, nem existe código mal estruturado. Até este ponto já conseguimos garantir uma maior *performance*, no entanto foram também implementadas outras medidas: a chamada aos serviços foi feita de forma a evitar constantes pedidos a estes (por exemplo, na submissão de novas ausências foi construido um pedido POST que permite enviar todas as novas ausências de uma vez, ao invés de se enviar uma nova ausência a cada pedido POST, resultando no menor tempo de espera por parte do utilizador); Sempre que existe um acesso aos serviços externos, é apresentado no ecrã a informação de que a aplicação está a processar dados (*loader*) – permitindo um maior *feedback* do que está a acontecer na aplicação; Para além disso foram usadas técnicas com recurso ao MvvmCross – *e.g.* foram usadas *RecyclerViews* que permitem que as listas tenham uma maior *performance* através da reutilização das instâncias das listas;

- **NF05** (Metodologia de desenvolvimento): como foi visto no Capítulo 3, foi usada uma metodologia ágil, que corresponde à metodologia usada dentro da empresa, permitindo uma maior organização das tarefas a realizar através da plataforma JIRA.

7.2 Xamarin e a reutilização de código

Na Capítulo 2 (Estado da Arte), foi referido que uma das grande forças da Xamarin é o facto de conseguir uma poupança de código de até cerca de 75%[1],para além de todas as outras vantagens já referidas. Para verificar que realmente esta alternativa cumpriu um dos seus propósitos, foi feito um cálculo da percentagem de código que está exatamente a ser utilizado em todo o projeto.

7.2.1 Metodologia e Resultados

Para calcular a taxa de código partilhado foram contabilizadas as linhas de código de cada um dos projetos (core, android e UWP) e de seguida foi comparada a proporção de código entre os projetos.

[1]Informação disponibilizada pela própria Xamarin no *website* deles, URL: `https://www.xamarin.com/platform` (acedido em Janeiro de 2017)

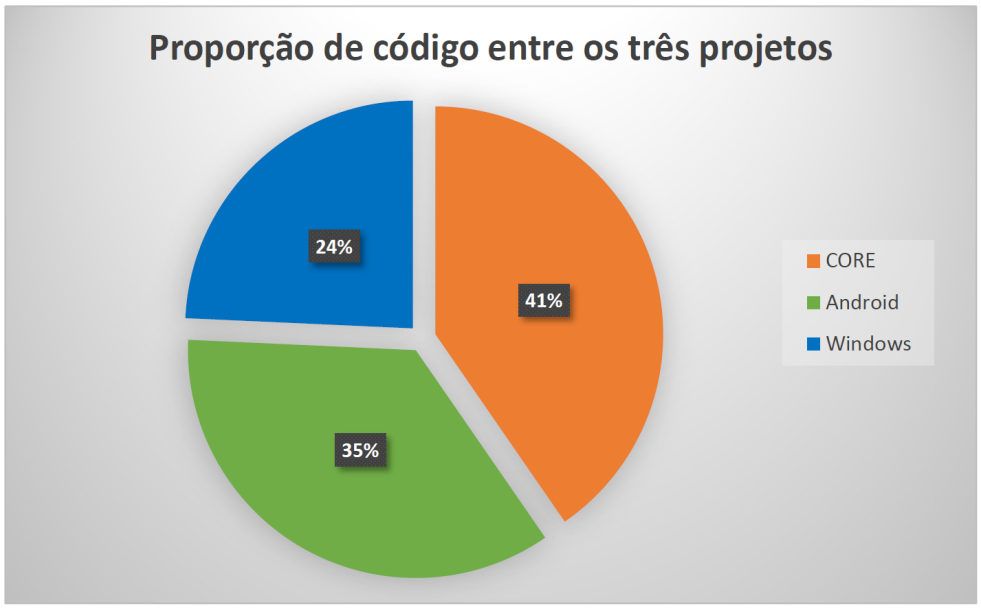

Figura 7.2: Partilha de código na Xpack

Os resultados foram evidentes como podemos ver na figura 7.2, conseguindo obter cerca de 41% de código partilhado entre as duas plataformas. Obtiveram-se os seguintes resultados:

- Core – 6587 linhas de código (41%);
- Android – 5765 linhas de código (35%);
- Windows – 3960 linhas de código (24%).

Apesar dos valores estarem longe dos 75%, termos 41% de código partilhado representa uma grande mais valia no desenvolvimento de um projeto de média ou grande dimensão. Além disso, os 75% referidos pela Xamarin partem do princípio que não serão feitos controlos / UI específicos. Quanto mais rica for a UI, menor é a reutilização de código.

Para além destes dados, e visto que foram recolhidos os dados relativamente aos três projetos, apresento na Fig. 7.3 as estatísticas do código do projeto Core (que partes do projeto conseguem uma maior dimensão). Como previsto, as *View-Models* representam mais de metade do código deste projeto (visto que são estas as responsáveis por transmitir diretamente as informações às plataformas) com cerca de 57%, seguido dos serviços com cerca de 29%, os modelos de dados com 13% e outros ficheiros de configuração e *converters* com os restantes 1%.

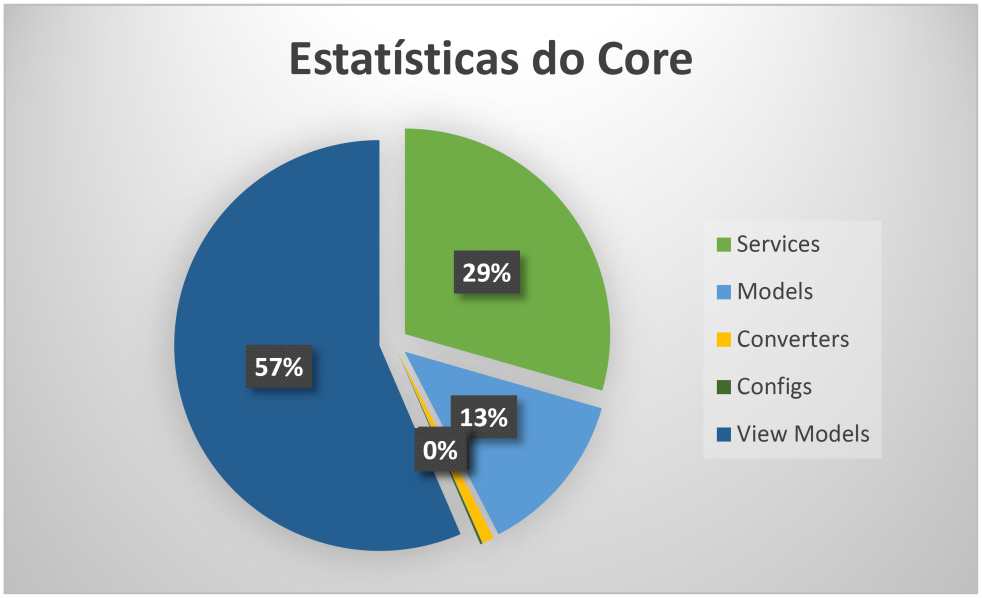

Figura 7.3: Estatísticas do projeto Core

7.3 Reflexão final e desenvolvimentos futuros

Em primeiro ponto, dado que já houve uma primeira *release* desta aplicação, houve também *feedback* por parte dos utilizadores sobre a Xpack. Assim, esta permitiu criar um canal de comunicação muito importante e essencial para a vida interpessoal e profissional da empresa. Através do lançamento das *Push Notifications*, vai haver um novo espaço para comunicar com os colaboradores de forma rápida, fácil e instantânea.

Como futuros desenvolvimentos, vão haver alterações a nível das *Push Notifications*, que irão permitir lançar a notificação push não só para todos os colaboradores, mas de forma individual (um ou mais colaboradores de forma isolada) ou em grupo (*e.g.*, por departamento). Para além disso, como pudemos ver pelos testes de usabilidade (secção 6.1), existem diversas melhorias que podem e devem ser feitas a nível de UI, para tornar esta aplicação mais intuitiva e fácil de usar.

Como nota, visto que o objetivo desta dissertação foi desenvolver a aplicação em Android e Windows, existiu a possibilidade de se desenvolver a aplicação também para iOS, que foi feito por outros colaboradores da empresa. Essa aplicação também se encontra desenvolvida devido ao facto de já existirem todas as infraestruturas (core e serviços) desenvolvidas no momento em que começaram a construir a aplicação iOS. Razão pela qual essa aplicação foi construida mais ra-

pidamente (salientando mais uma vez as vantagens da Xamarin).

Quer tudo isto dizer que este não é um projeto para o qual não se preveja evolução. Existe espaço para melhorias e criação de novas funcionalidades para tornar este produto cada vez mais imprescindível na vida profissional dos colaboradores da Xpand IT. Como extra, no futuro a intenção é conseguir tornar esta aplicação flexível e inovadora a ponto de vender este produto a outras empresas.

Bibliografia

[1] Abozar AlizadehAbozar Alizadeh. *Xamarin, Cons Pros*. URL: https://www.linkedin.com/pulse/xamarin-cons-pros-abozar-alizadeh (acedido em 01/06/2016).

[2] *Appcelerator Titanium*. URL: http://www.appcelerator.com/ (acedido em 03/04/2016).

[3] John Bristowe. *What is a Hybrid Mobile App?* URL: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/ (acedido em 27/03/2016).

[4] *Cross compiler*. URL: https://en.wikipedia.org/wiki/Cross_compiler (acedido em 19/03/2016).

[5] Andrea Dallera. *Why you should stay away from Appcelerator's Titanium*. URL: https://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerators-titanium/ (acedido em 09/05/2016).

[6] *Developer Economics: State of Developer Nation Q1 2016*. URL: https://www.developereconomics.com/reports/developer-economics-state-of-developer-nation-q1-2016/ (acedido em 12/04/2016).

[7] Nat Friedman. *A Xamarin + Microsoft Future*. URL: https://blog.xamarin.com/a-xamarin-microsoft-future/ (acedido em 30/02/2016).

[8] T. A. Majchrzak. H. Heitkötter S. Hanschke. "Evaluating cross-platform development approaches for mobile applications". Em: *Lecture Notes in Business Information Processing* Vol. 140 (2013), pp. 120–138.

[9] *Introduction to Xamarin Test Cloud*. URL: https://developer.xamarin.com/guides/testcloud/introduction-to-test-cloud/ (acedido em 12/05/2016).

[10] Francisco Iury. *Desenvolvimento de aplicações móveis híbridas com Sencha Touch 2*. URL: http://www.devmedia.com.br/desenvolvimento-de-aplicacoes-moveis-hibridas-com-sencha-touch-2/26150 (acedido em 29/04/2016).

[11] Maitrik Kataria. *Native vs cross platform development – Performance limitations*. URL: https://www.simform.com/blog/native-vs-cross-platform-developme (acedido em 18/03/2016).

[12] Stank Kemp. *Why Appcelerator Titanium Considers The Smartest Framework?* URL: https://www.linkedin.com/pulse/20140419110331-192789735-why-appcelerator-titanium-considers-the-smartest-framework (acedido em 02/05/2016).

[13] Thomas LeBrun. *Windows Phone : Build MVVM Apps with Xamarin and MvvmCross*. URL: https://msdn.microsoft.com/en-us/magazine/dn759442.aspx (acedido em 25/03/2016).

[14] *Mobile Apps in Visual Studio with Xamarin*. URL: https://www.visualstudio.com/en-us/features/xamarin-vs.aspx (acedido em 16/04/2016).

[15] *PhoneGap*. URL: https://pt.wikipedia.org/wiki/PhoneGap (acedido em 02/04/2016).

[16] *Pros and Cons of Developing Native vs. Cross-Platform Web-Based Mobile Application*. URL: https://www.dbbest.com/blog/pros-and-cons-of-developing-native-vs-cross-platform-web-based-mobile-application/ (acedido em 01/03/2016).

[17] *Sencha Touch*. URL: https://www.sencha.com/products/touch/#overview (acedido em 06/04/2016).

[18] *The Pros and Cons of Native Apps and Mobile Web Apps*. URL: http://mobiledevices.about.com/od/additionalresources/qt/The-Pros-And-Cons-Of-Native-Apps-And-Mobile-Web-Apps.htm (acedido em 01/03/2016).

[19] *When to Go Native, Cross-Platform and Hybrid for Mobile*. URL: http://www.program-ace.com/press-room/articles/native-cross-platform-hybrid-for-mobile (acedido em 01/03/2016).

[20] S. Xanthopoulos e S. Xinogalos. "A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications". Em: *Proceedings of the 6th Balkan Conference in Informatics* (2013), pp. 213–220. URL: http://doi.acm.%20org/10.1145/2490257.2490292.

Apêndice A

Casos de uso

Ator	Colaborador
Prioridade	Must
Pré-condições	• Não ter sessão iniciada
Pós-condições	• Login efetuado • É possível usufruir das funcionalidades da app

Tabela A.1: UC01 - Login

Ator	Colaborador
Prioridade	Should
Pré-condições	• Não ter sessão iniciada
Pós-condições	• Ao ativar esta opção não será necessário introduzir credenciais novamente a não ser que faça explicitamente "Logout"

Tabela A.2: UC02 - Lembrar credenciais

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado
Pós-condições	• É possível visualizar a lista de notícias • É possível ver o detalhe de uma notícia

Tabela A.3: UC03 - Aceder à Homepage

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã "Homepage"
Pós-condições	• É possível visualizar uma notícia completa

Tabela A.4: UC04 - Ver detalhe de uma notícia

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã de "Homepage"ou "Apps"
Pós-condições	• É possível aceder à aplicação JIRA através de uma *webview*

Tabela A.5: UC05 - Iniciar JIRA

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã de "Homepage"ou "Apps"
Pós-condições	• É possível aceder à aplicação Confluence através de uma *webview*

Tabela A.6: UC06 - Iniciar Confluence

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã de "Homepage"ou "Apps"
Pós-condições	• É possível aceder à aplicação Confluence através de uma *webview*

Tabela A.7: UC07 - Iniciar Service Desk

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado
Pós-condições	• É possível aceder às aplicações JIRA, Confluence e Service Desk

Tabela A.8: UC08 - Ver lista de aplicações

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado
Pós-condições	• É possível aceder à lista dos contactos de todos os colaboradores da empresa

Tabela A.9: UC09 - Ver lista de contactos

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Lista de contactos"
Pós-condições	• É possível aceder aos contactos de um colaborador (telefone, mail, skype, etc. • É possível realizar uma chamada, ou enviar uma mensagem por telemóvel ao colaborador

Tabela A.10: UC10 - Ver lista de contactos

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã de detalhe de um contacto
Pós-condições	• É possível adicionar o contacto do colaborador à lista de contactos do telemóvel

Tabela A.11: UC11 - Adicionar contacto ao telemóvel

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã de detalhe de um contacto
Pós-condições	• É possível realizar uma chamada através do telemóvel ao colaborador

Tabela A.12: UC12 - Ligar ao contacto

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã de detalhe de um contacto
Pós-condições	• É possível enviar uma mensagem de texto através do telemóvel ao colaborador

Tabela A.13: UC13 - Enviar mensagem ao contacto

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado
Pós-condições	• É possível aceder à informação fiscal da Xpand IT em Portugal e em Inglaterra

Tabela A.14: UC14 - Aceder à informação fiscal da Xpand IT

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado
Pós-condições	• É possível enviar a informação dos contactos do colaborador autenticado para outro contacto

Tabela A.15: UC15 - Enviar contacto

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã "enviar contacto"
Pós-condições	• É possível enviar o contacto através de um Vcard

Tabela A.16: UC16 - Enviar Vcard

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã "Enviar contacto"
Pós-condições	• É possível enviar o contacto através de um Mail

Tabela A.17: UC17 - Enviar Email

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado
Pós-condições	• É possível aceder à morada dos três escritórios da Xpand IT

Tabela A.18: UC18 - Aceder à morada dos escritórios

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado • Estar no ecrã "Morada dos escritórios"
Pós-condições	• É possível aceder à morada do escritório de Viana do Castelo

Tabela A.19: UC19 - Morada de Viana

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado • Estar no ecrã "Morada dos escritórios"
Pós-condições	• É possível aceder à morada do escritório de Lisboa

Tabela A.20: UC20 - Morada de Lisboa

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado • Estar no ecrã "Morada dos escritórios"
Pós-condições	• É possível aceder à morada do escritório de Londres

Tabela A.21: UC21 - Morada de Londres

Ator	Colaborador
Prioridade	Cloud
Pré-condições	• Login efetuado
Pós-condições	• É possível aceder às informações do colaborador • É possível alterar as credenciais • É possível adicionar um novo número de telefone

Tabela A.22: UC22 - Editar perfil

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Lista de aplicações"
Pós-condições	• É possível visualizar a lista das timesheets por semana • É possível navegar entre as diferentes semanas • É possível ver o estado da timesheet de uma dada semana

Tabela A.23: UC23 - Àceder às Timesheets

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado • Estar no ecrã "Timesheets"
Pós-condições	• É possível copiar as timesheets de uma dada semana para outra de formal integral

Tabela A.24: UC24 - Copiar semana

Ator	Colaborador
Prioridade	Could
Pré-condições	• Login efetuado • Estar no ecrã "Timesheets"
Pós-condições	• É possível copiar as entradas de um dado dia ao fazer swipe da esquerda para a direita sobre o dia a copiar, e a colar, respetivamente

Tabela A.25: UC25 - Copiar dia

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Timesheets"
Pós-condições	• As timesheets são guardadas no servidor mas não submetidas

Tabela A.26: UC26 - Gravar Timesheets

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Timesheets"
Pós-condições	• A timesheet da semana selecionada é gravada no servidor e fica em estado submetida, esperando por aprovação

Tabela A.27: UC27 - Submeter Timesheets

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Timesheets"
Pós-condições	• As entradas do dia selecionado são apresentadas no ecrã com as informações base desta

Tabela A.28: UC28 - Visualizar entradas de um dia

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Entradas"
Pós-condições	• São apresentadas as informações da entrada (caso tenha sido selecionada uma entrada no ecrã anterior) • É possível adicionar uma nova entrada caso a timesheet desse dia ainda não esteja submetida • No caso de ter sido selecionada uma entrada que ainda não foi submetida, é possível editar esta

Tabela A.29: UC29 - Adicionar ou editar entrada

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado
Pós-condições	• É visualizado um mapa anual em que os meses com férias marcadas estão visíveis como tal • É possível navegar entre os vários anos • Ao rodar o ecrã do dispositivo, é possível ver um gráfico de barras com as ausências quantificadas por mês

Tabela A.30: UC30 - Aceder ao mapa de ausências

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Mapa de ausências"
Pós-condições	• É possível marcar um intervalo de dias com uma determinada ausência e gravar

Tabela A.31: UC31 - Alterar o mapa de ausências

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado
Pós-condições	• É possível visualizar todas as informações de perfil do colaborador

Tabela A.32: UC32 - Aceder ao perfil

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado • Estar no ecrã "Perfil"
Pós-condições	• Caso as credenciais estejam corretas, a palavra passe é modificada com sucesso em todo o sistema dentro da empresa.

Tabela A.33: UC33 - Alterar palavra passe

Ator	Colaborador
Prioridade	Should
Pré-condições	• Login efetuado • Estar no ecrã "Perfil"
Pós-condições	• É adicionado um número de telefone.

Tabela A.34: UC34 - Adicionar número de telefone

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado com privilégios de administrador.
Pós-condições	• É enviada uma notificação push para todos os colaboradores da empresa.

Tabela A.35: UC35 - Enviar notificações push

Ator	Colaborador
Prioridade	Must
Pré-condições	• Login efetuado
Pós-condições	• Logout efetuado • É necessário fazer uma nova autenticação para ter acesso às funcionalidades da app • As alterações são refletidas no ecrã anterior

Tabela A.36: UC36 - Logout

Apêndice B

Design final

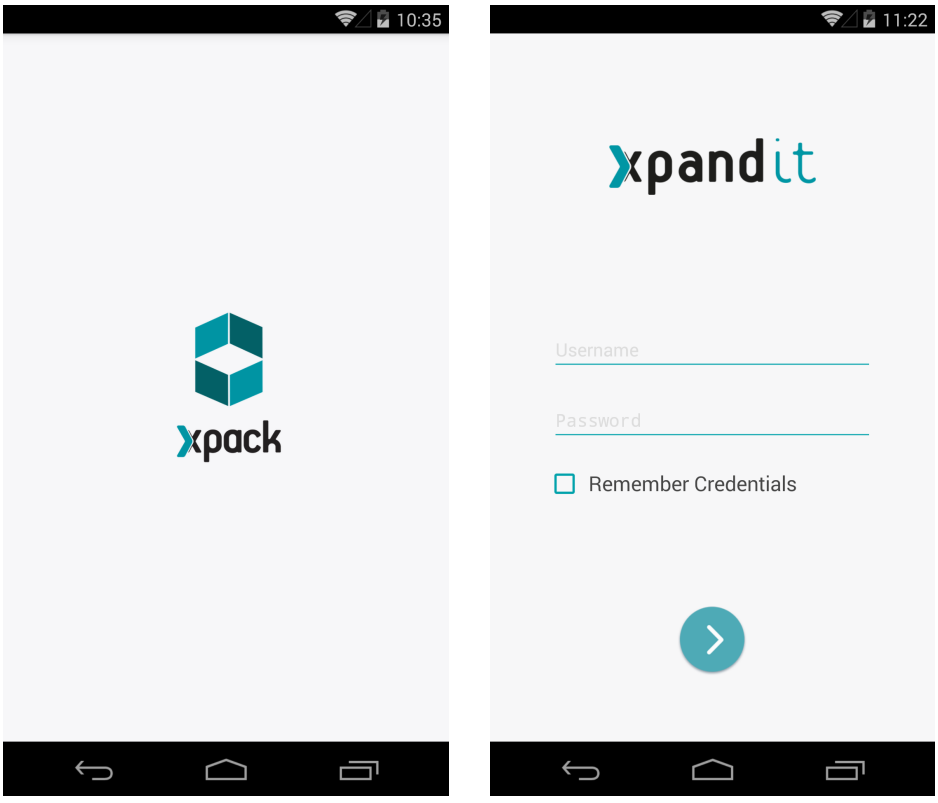

Figura B.1: Splash e Login Screen

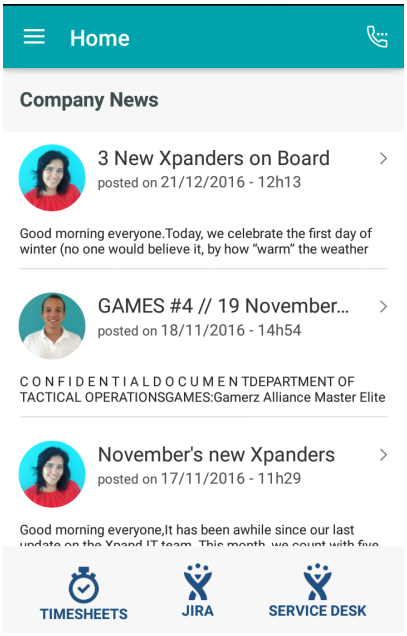

Figura B.2: Home

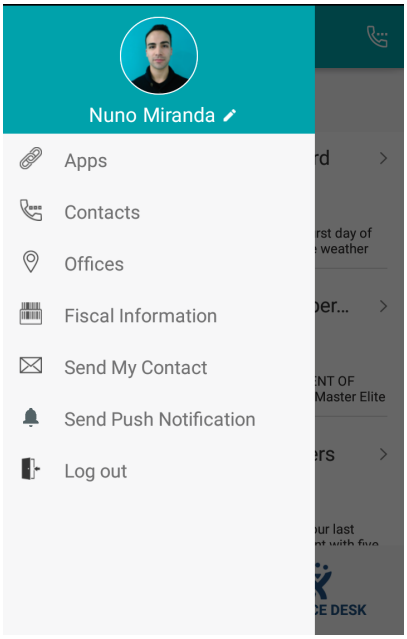

Figura B.3: Menu

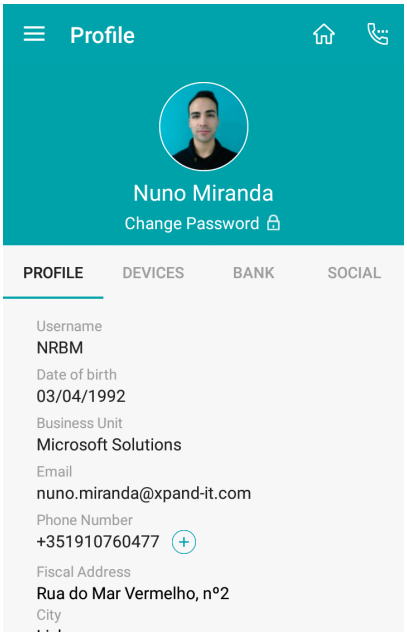

Figura B.4: Editar Perfil

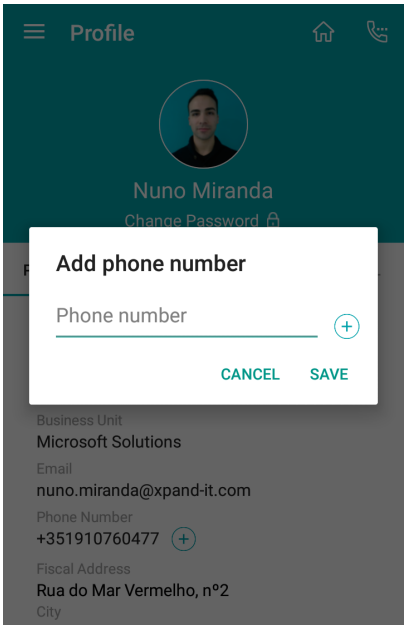

Figura B.5: Editar Perfil: adicionar telefone

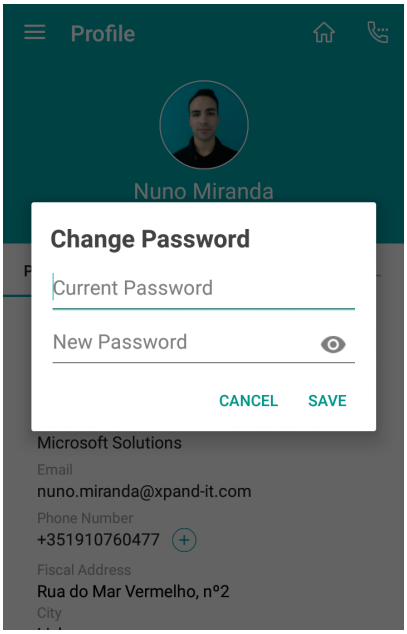

Figura B.6: Editar Perfil: mudar palavra-passe

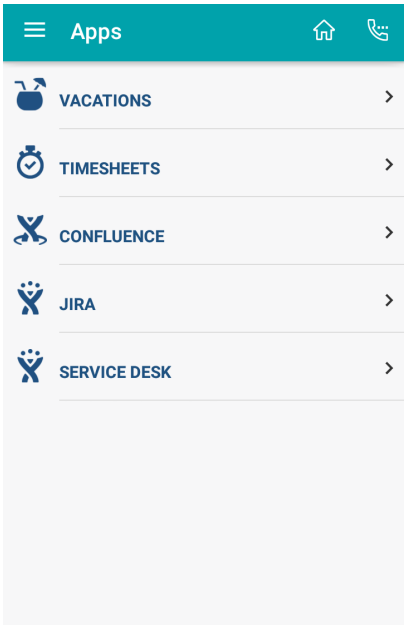

Figura B.7: Apps

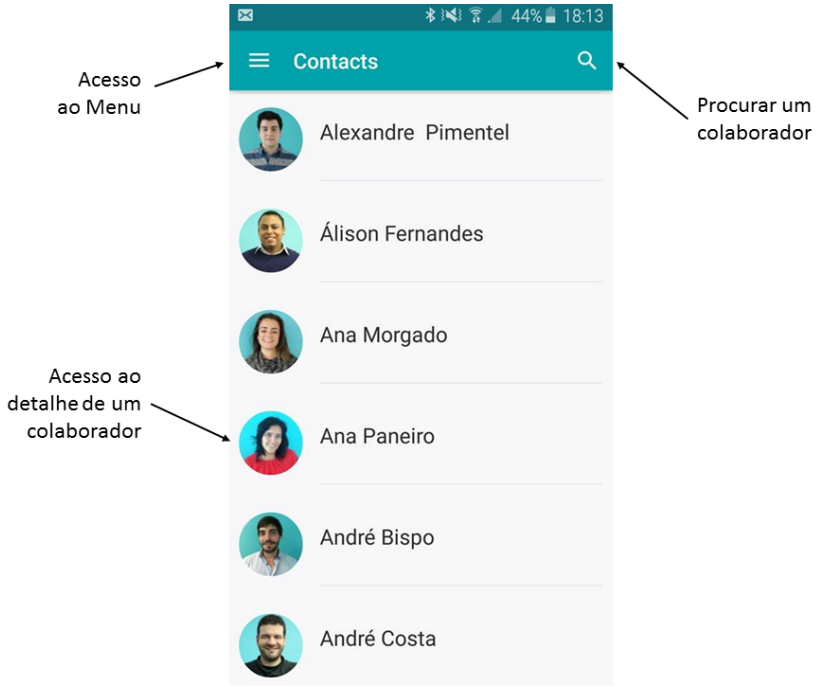

Figura B.8: Contactos

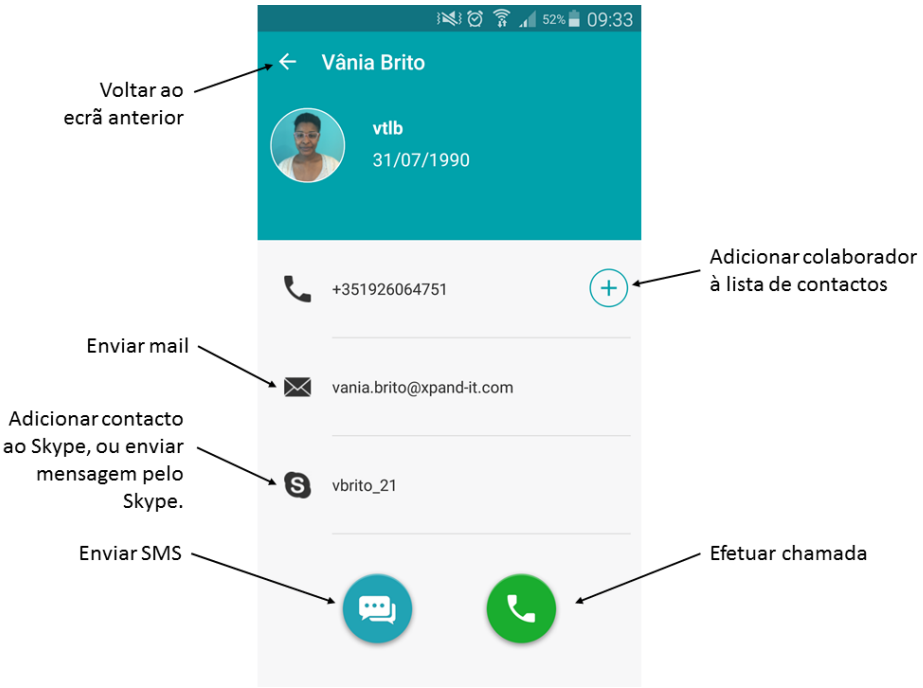

Figura B.9: Contactos (detalhe de contacto)

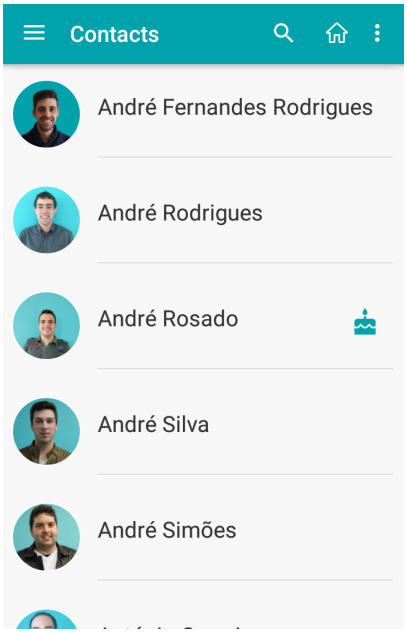

Figura B.10: Aniversário de um contacto na lista de contactos

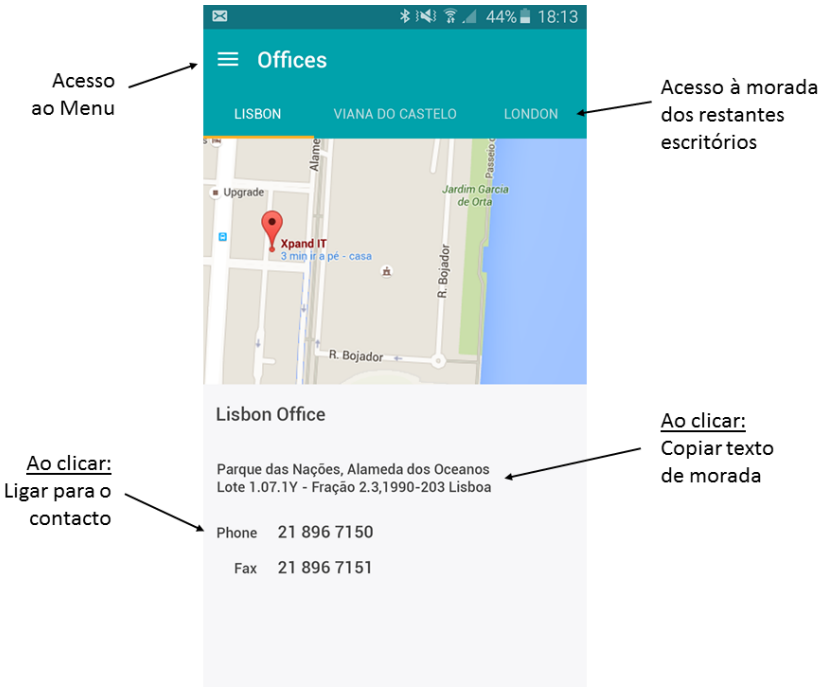

Figura B.11: Escritórios

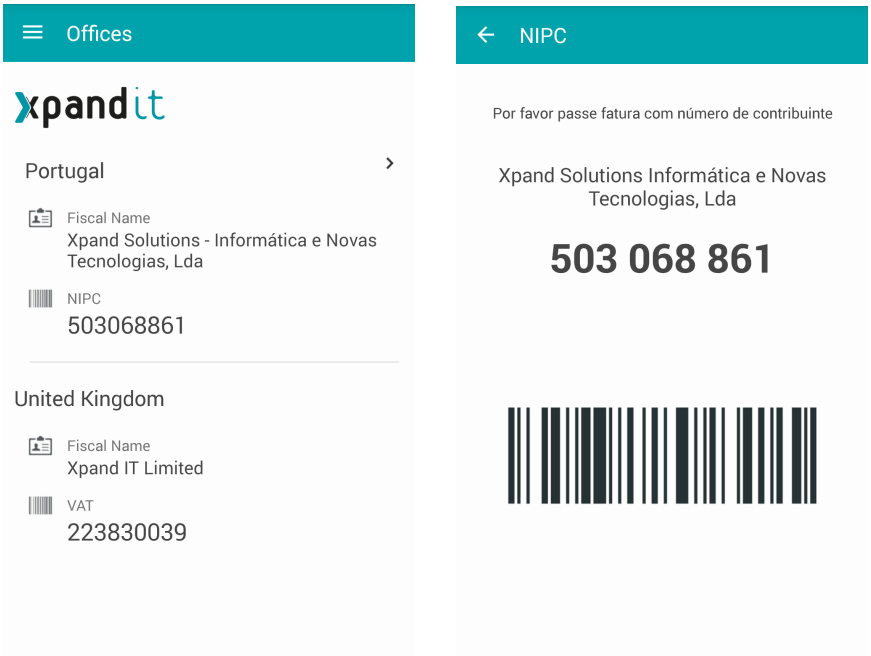

Figura B.12: Informação fiscal da Xpand IT

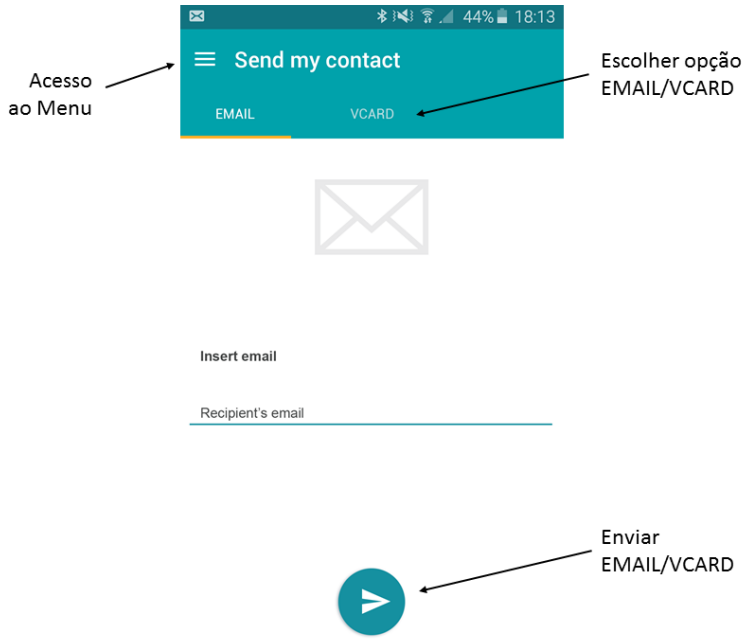

Figura B.13: Enviar contacto por email

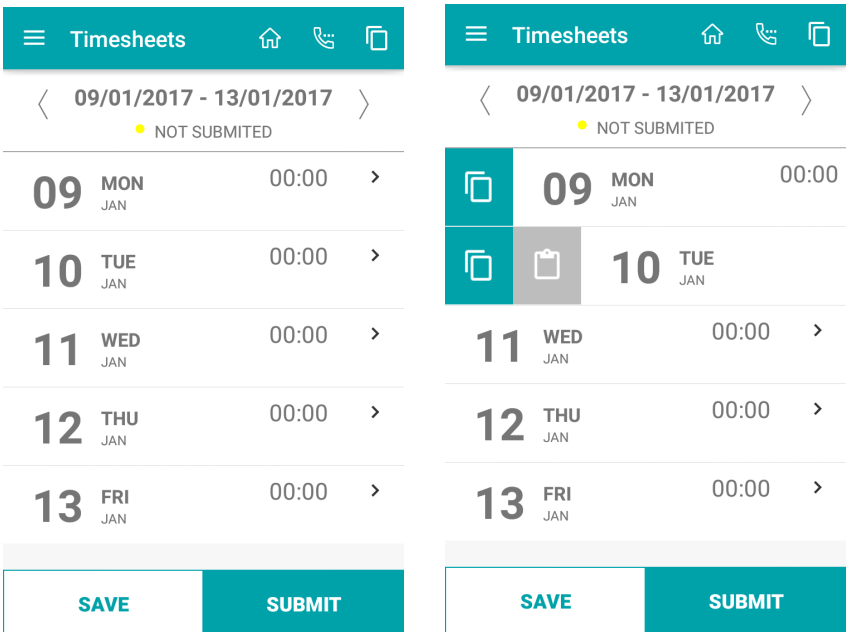

Figura B.14: Timesheets - Ecrã principal

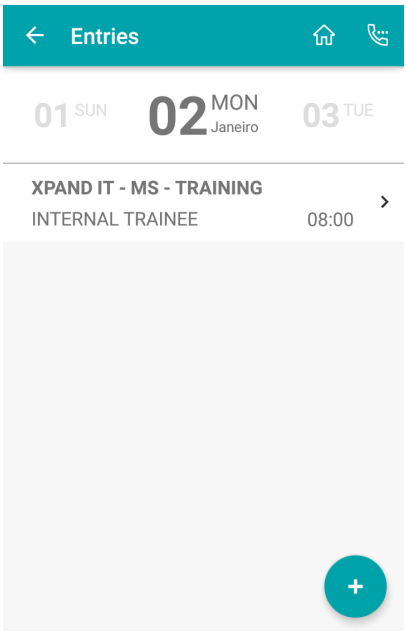

Figura B.15: Timesheets - Entradas

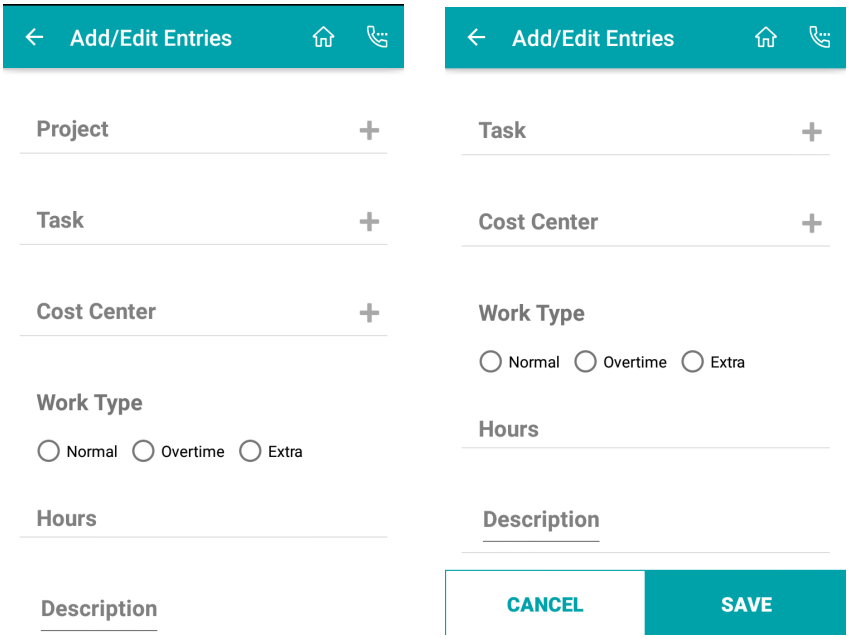

Figura B.16: Timesheets - Adicionar nova entrada

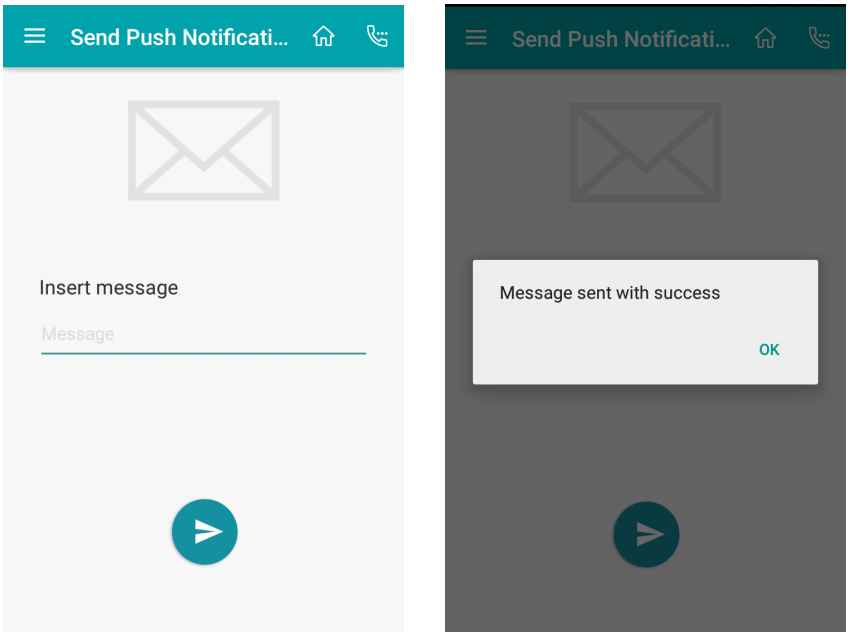

Figura B.17: Enviar notificação push

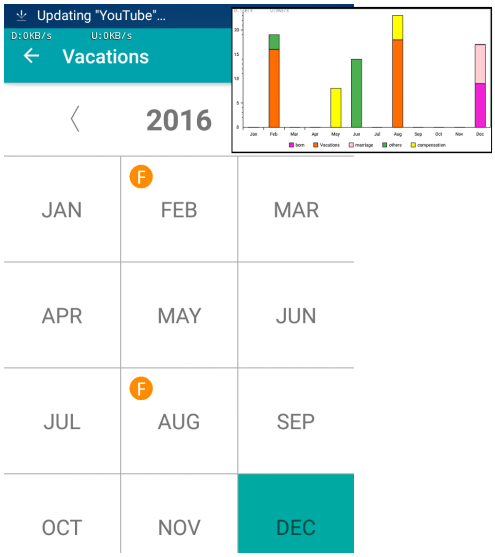

Figura B.18: Ausências: visão anual

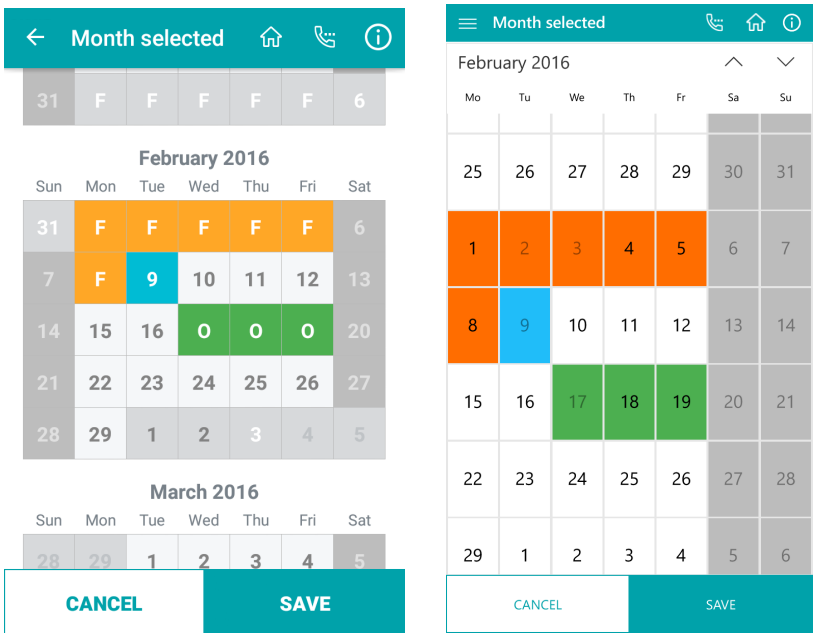

Figura B.19: Ausências: visão mensal (Android e UWP, respetivamente)

Figura B.20: Ecrã de ausências: visão mensal

Apêndice C

Resultados dos testes de aceitação

Nas tabelas seguintes estão apresentados os resultados de cada um dos testes. Os identificadores "I01, I02, .. , I06"correspondem a cada um dos inquiridos. Foi também pedido a cada um dos inquiridos no final de cada tarefa, que avaliasse o grau de dificuldade dessa tarefa de 1 a 10, sendo que 1 representa "Tarefa extremamente difícil de executar"e 10 representa "Tarefa extremamente fácil de executar". Foi usada esta escala para evitar casos em que os inquiridos escolhessem "o meio"da escala.

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	7	3	3	3	3	4
Tempo	01:55	00:33	00:28	00:27	00:28	00:34
Grau de dificuldade	10	10	10	10	10	10
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.1: Dados relativos à Tarefa T01

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	12	3	5	5	4	6
Tempo	02:05	00:11	00:31	00:19	00:13	00:21
Grau de dificuldade	5	10	9	10	10	10
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.2: Dados relativos à Tarefa T02

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	3	3	4	3	3	4
Tempo	00:10	00:10	00:11	00:06	00:08	00:09
Grau de dificuldade	5	10	9	10	10	10
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.3: Dados relativos à Tarefa T03

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	10	5	5	9	5	5
Tempo	00:54	00:32	00:38	00:25	00:28	00:35
Grau de dificuldade	9	10	9	10	9	10
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.4: Dados relativos à Tarefa T04

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	16	15	11	12	11	13
Tempo	01:33	01:28	00:56	00:33	01:12	00:53
Grau de dificuldade	6	6	5	8	5	6
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.5: Dados relativos à Tarefa T05

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	22	16	19	19	14	18
Tempo	03:48	01:40	02:13	01:20	01:11	01:58
Grau de dificuldade	7	9	9	7	7	8
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.6: Dados relativos à Tarefa T06

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	10	46	13	29	19	23
Tempo	01:48	05:07	01:02	01:08	01:31	01:45
Grau de dificuldade	1	1	1	1	1	1
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.7: Dados relativos à Tarefa T07

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	29	10	8	9	7	12
Tempo	02:46	00:58	00:15	00:32	00:28	00:25
Grau de dificuldade	1	10	10	10	9	9
Tarefa concluída com sucesso	não	sim	sim	sim	sim	sim

Tabela C.8: Dados relativos à Tarefa T08

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	14	7	21	7	17	19
Tempo	02:50	00:32	02:45	00:35	01:06	01:29
Grau de dificuldade	8	10	9	10	7	8
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.9: Dados relativos à Tarefa T09

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	15	13	7	23	7	8
Tempo	00:30	01:40	01:28	02:27	01:20	01:57
Grau de dificuldade	8	8	9	7	10	8
Tarefa concluída com sucesso	parcialmente	sim	sim	sim	sim	sim

Tabela C.10: Dados relativos à Tarefa T10

	I01	**I02**	**I03**	**I04**	**I05**	**I06**
Número de cliques	2	2	2	2	2	2
Tempo	00:18	00:05	00:05	00:06	00:05	00:05
Grau de dificuldade	10	10	10	10	10	10
Tarefa concluída com sucesso	sim	sim	sim	sim	sim	sim

Tabela C.11: Dados relativos à Tarefa T11