UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Desenvolvimento de aplicação móvel e serviços de suporte

Tiago Alexandre Fernandes de Noronha

Mestrado em Engenharia Informática
Sistemas de Informação

Trabalho de Projeto orientado por:
João Balsa da Silva e Francisco Ceia

2017

Agradecimentos

À minha namorada que está sempre presente, a incentivar-me em todas as alturas.

À minha mãe que faz tudo para que tenha sempre o maior sucesso, sem nunca esperar nada em troca.

Ao meu pai que sem ele nunca chegaria onde cheguei, nem teria seguido o caminho que segui.

Aos meus avós que estão sempre prontos a ajudar.

Ao meu primo Hugo que me ajudou a tomar a decisão certa na vida académica, tal como aos meus tios e primos que sempre que apoiaram.

Ao meu orientador o Professor João Balsa por me auxiliar durante todo o curso.

Ao meu orientador Francisco Ceia por m

e ajudar a envergar no mundo do trabalho, estando constantemente a abrir portas para novos desafios, tal como ao Nuno Quintas e ao André Dias, a quem “roubava” sempre tempo para me ajudarem.

À Inês Ribeiro por me ajudar a integrar desde o dia em que cheguei à empresa.

À Sofia Gonçalves, à Ana Châtillon e à Catarina Ribeiro que corrigiram as minhas calinadas.

A todos os membros da Unipartner pelo excelente ambiente no local de trabalho.

A todos que me ajudaram direta ou indiretamente na realização do projeto de mestrado, o meu obrigado.

Para a Pat.

Resumo

Muito do crescimento do mercado *mobile* deve-se ao aumento da exigência dos utilizadores. Para dar resposta a estes novos requisitos são necessárias ferramentas que produzam soluções abrangentes, de modo a chegar ao maior número de mercados possível, e rápidas, de forma a dar resposta às necessidades dos utilizadores com maior brevidade possível. Surge então o conceito de *cross-platform*, que consiste no desenvolvimento de aplicações de forma a serem usados em vários sistemas operativos.

Este relatório descreve o trabalho realizado na criação de uma aplicação *cross-platform* ao longo do estágio na empresa Unipartner IT Services S. A., daqui para a frente referida apenas como Unipartner. Destaca os conceitos mais importantes, as tecnologias e metodologias utilizadas durante o projeto, assim como a motivação para a resolução dos problemas descritos na introdução. Refere o trabalho relacionado com o projeto, expõe o trabalho realizado e os desafios encontrados no decorrer do desenvolvimento, e por fim defende as escolhas tomadas durante as fases de desenho e desenvolvimento, e respetivas consequências.

Palavras-chave: *Cross-platform*; Azure; Desenvolvimento móvel; Xamarin.

Abstract

Much of the growth of the mobile development market is due to the uprising demands of the users. It's now necessary to have tools which produce broader solutions to answer these new requirements, so that the needs of the users are fulfilled as fast as possible. The concept of Cross-platform arrives, which is described as the development of applications to be used in multiple operating systems.

This report describes the work done while creating a cross-platform application, along the internship at Unipartner IT Services S. A (from now on referred to as "Unipartner"). It focuses on the most important concepts, the technology and the methodologies addressed during the project as well as the motivation to resolve the problems described at the introduction. The related work is addressed, as well as the work done and the challenges encountered. Lastly it will explain the choices made during the development process and the consequences that arrived from those choices.

Keywords: Cross-platform, Azure, Mobile Development, Xamarin

Conteúdo

Acrónimos

APNS - Apple Push Notification Service

CESD - Cartão Europeu de Saúde e Doença

CRUD - Create, Read, Update, Delete

DI - Dependency Injection

FCM - Firebase Cloud Messaging

IaaS - Infrastructure as a Service

IoC - Inversion of Control

MitM - Man in the Middle

MVC - Model View Controller

MVVM - Model View ViewModel

NSG - Network Security Group

PaaS - Platform as a Service

PCL - Portable Class Library

PNS - Platform Notification Service

UI - User Interface

UWP - Universal Windows Platform

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo é dado um contexto do projeto, descrita a instituição de acolhimento do estágio, tal como a motivação, a contribuição e os objetivos do projeto. É também referida a organização do documento.

1.1 Contexto

O mundo tornou-se *mobile*. Cada vez mais os *smartphones* começam a ser algo indispensável no nosso dia-a-dia. Os dispositivos móveis já permitem aos utilizadores fazer tudo no momento, seja tomar notas, planear viagens, captar momentos e até possibilitam uma conexão imediata com os nossos familiares ou colegas, independentemente do local onde se encontram.

Não é fácil imaginar fazer tarefas, que hoje consideramos triviais, antes da tecnologia que temos disponível aparecer, por exemplo descobrir o caminho para algum local desconhecido, ou até encontrar caminhos alternativos, como faríamos sem um GPS dentro do bolso.

É raro hoje encontrar um telemóvel que não permita a conexão à Internet. A facilidade enorme na obtenção de um *smartphone* com capacidade de acesso à internet traduz-se numa aposta cada vez maior de disponibilização de conteúdos em formato mobile. Esta difusão acontece pelo fato da informação poder ser acedida de forma mais ágil, devido à evolução das redes de comunicação (3G, 4G, etc), e por permitirem ao usuário estar informado a qualquer momento e em qualquer lugar. Desta forma as organizações conseguem aproximar-se dos utilizadores, criando ao mesmo tempo um canal, à partida sempre disponível, que disponibiliza informação imediata.

Por outro lado, a diversidade de sistemas operativos e hardware traz desafios ao nível do desenvolvimento de aplicações móveis. Para tal, e para conseguir suportar o maior número possível de *smartphones* e assim chegar a um maior número de

utilizadores, é necessário adaptar o nosso código. Com vista a colmatar esse problema surgiu o conceito de *cross-platform*, descrito no decorrer deste projeto.

1.2 A Unipartner

O projeto decorre na Unipartner, empresa nacional que foi criada em 2015 após uma reorganização da vertente de negócio da antiga Unisys Portugal, que fez com que esta se retirasse do nosso país. A Unipartner concede serviços de consultoria nas áreas de *cloud* e infraestruturas, desenvolvimento aplicacional e gestão de projetos, sendo uma casa muito orientada para as tecnologias Microsoft. Foi nesta empresa que decorreu o projeto sobre o qual este documento incide, que tem como destinatário um cliente da Unipartner.

1.2.1 A equipa

O projeto foi inserido numa equipa multidisciplinar, com diferentes níveis de senioridade. Essa equipa é composta por pessoas mais experientes com vasta abrangência de desafios similares, tendo todos experiência em programação. A proximidade entre a Unipartner e o cliente, sendo este um parceiro no atingimento dos objetivos descritos, permitiu então obter uma relação benéfica para ambas as equipas envolvidas que se traduziu numa melhor comunicação e discussão de tomadas de decisão.

1.3 Motivação

O cliente desenvolve a sua atividade orientada exclusivamente para o financiamento. Face ao atual enquadramento e opções tomadas, enfrenta agora um novo paradigma: assegurar a sustentabilidade do sistema. Para tal, é fundamental assegurar o permanente alinhamento entre as necessidades dos seus utentes e a oferta dos serviços de qualidade dos seus prestadores de serviços, promovendo a redução dos impactos e aumento dos benefícios gerados. Torna-se então essencial implementar medidas que permitam tirar melhor partido de efeitos de escala, aumentando a base de utilizadores e estendendo o leque de serviços prestados.

Assim sendo, a aposta no canal móvel surge como algo natural, desejável e estratégico de forma a consolidar globalmente este momento de modernização e de aproximação da organização ao utente. A disponibilização de aplicações móveis visa a otimização da experiência de relacionamento descrita acima, tirando partido da riqueza

do meio de comunicação e do tipo de funcionalidades oferecidas para exponenciar quer a forma como a informação é partilhada, quer o relacionamento de proximidade.

1.4 Objetivos

O maior objetivo deste projeto é a criação de uma aplicação móvel para o cliente, sendo o público-alvo os utentes do cliente, assegurando uma resposta efetiva às necessidades identificadas, bem como a modernização dos serviços de *back-end*, recorrendo a tecnologias mais modernas e altamente escaláveis, assegurando assim a sustentabilidade do sistema. Para atingir este objetivo foi produzida uma solução *cross-platform*, com um *back-end* sustentado na plataforma *cloud* da Microsoft - o Microsoft Azure.

1.5 Contribuições

Com este projeto foi desenvolvida uma aplicação móvel que não só permite uma maior aproximação da organização aos seus utilizadores, mas também uma melhoria na qualidade dos serviços oferecidos, transmitindo uma clara perceção de transformação e modernidade.

Toda a plataforma é continuamente medida e monitorizada, algo que suporta também a tomada de decisão e conduz a evolução do próprio negócio da organização ao disponibilizar serviços mais alinhados com as reais necessidades dos seus utentes.

Este projeto contribui também para sustentar toda a estratégia de mobilização dos vários serviços deste cliente, ao lançar as fundações para suportar todas as infraestruturas que consigam contemplar as necessidades atuais e futuras da organização.

1.6 Organização do documento

O relatório está dividido em 7 capítulos:

- O Capítulo 1 introduz o projeto, falando sobre a empresa de acolhimento, a motivação para a realização do projeto, as contribuições do mesmo e os objetivos.
- O Capítulo 2 apresenta as metodologias e conceitos utilizados durante o projeto.
- O Capítulo 3 detalha as tecnologias utilizadas no decorrer do projeto.
- No Capítulo 4 é descrito o trabalho relacionado, nomeadamente a investigação feita ao nível de *frameworks* alternativas, bem como o enquadramento do projeto.

- No Capítulo 5 é detalhado o trabalho realizado do lado da aplicação.
- No Capítulo 6 é detalhado o trabalho realizado no *back-end*.
- Finalmente o Capítulo 7 apresenta um resumo do projeto, tal como propõe trabalho futuro. É também feito um olhar crítico ao projeto, e referido o trabalho realizado em projetos não relacionados.

Capítulo 2

Metodologias e Conceitos

As metodologias e conceitos utilizados neste projeto e descritos neste capítulo permitem-nos desenhar e implementar a solução utilizando critérios e métodos rigorosos existentes no mercado, garantindo a eficiência, eficácia e qualidade da solução.

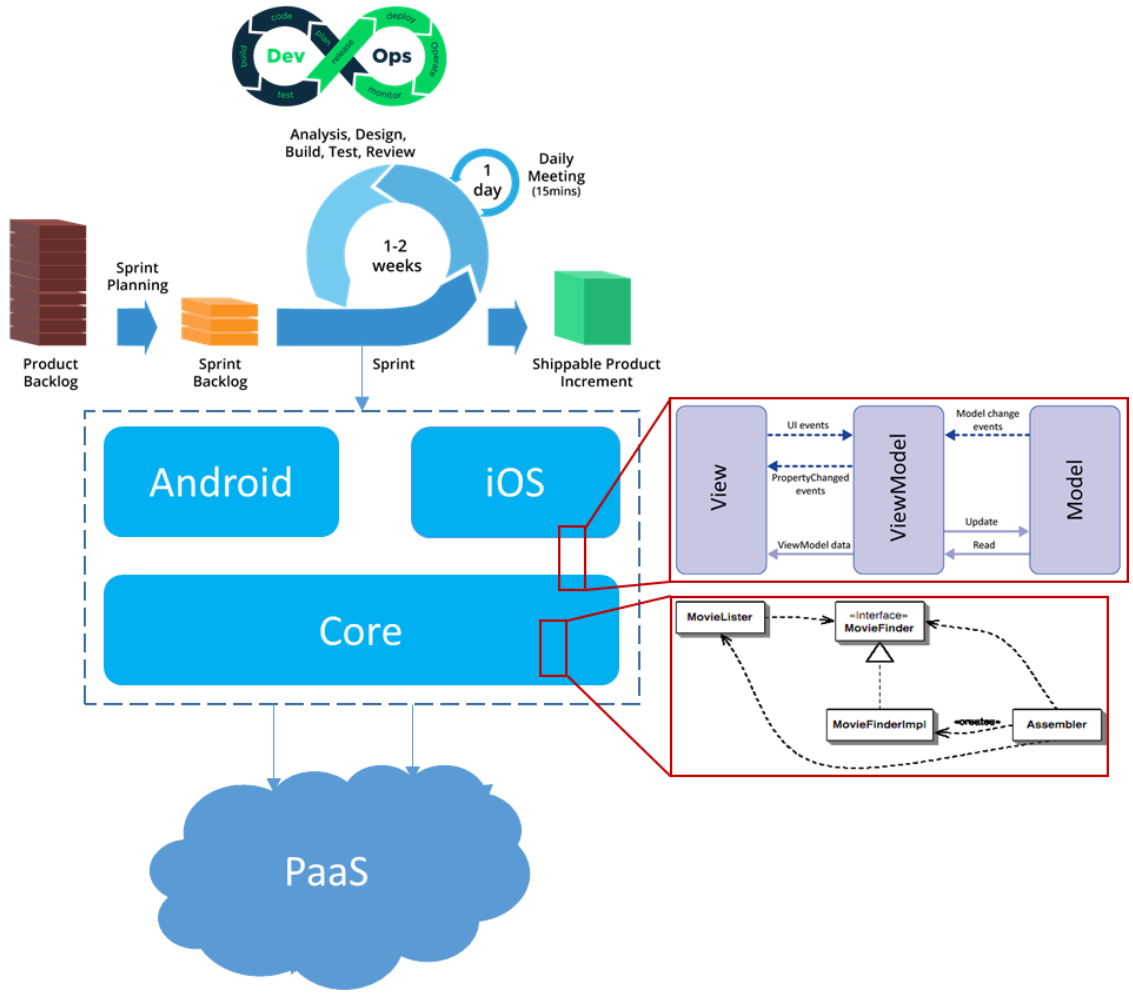

Figura 2.1 - Interligação das metodologias e dos conceitos

2.1 Metodologia Agile

Agile [1, 2] é um conjunto de princípios iterativos para desenvolvimento de *software*, introduzido para combater o método de desenvolvimento sequencial onde o cliente pode inspecionar o produto ao longo de "iterações", oferecendo feedback, ao invés de apenas ver o produto final após um período de tempo mais largo.

Esta abordagem permite também detetar e corrigir erros de implementação ou revisão de requisitos mais cedo, permitindo também que a falha, a acontecer, ocorra o mais cedo possível, havendo margem para experimentação e ajustes.

2.1.1 Scrum

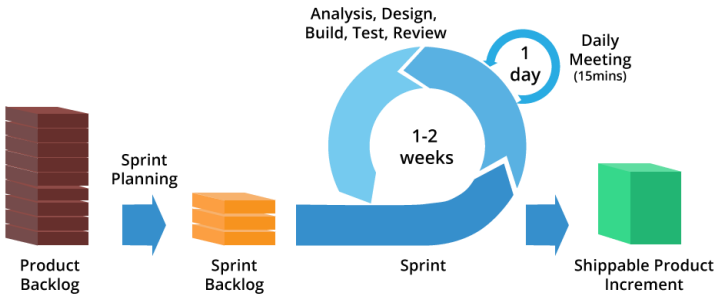

Figura 2.2 – Scrum

Scrum [3, 4] é uma *framework* de Agile. Enquanto que o Agile é um conjunto de princípios, o *Scrum* é uma implementação concreta.

Primariamente define-se os requisitos do produto, representado na Figura 2.2 pelo "*Product Backlog*". [5] Consoante os requisitos, planeia-se "*Sprints*", ou seja, iterações para desenvolver parte desses requisitos. No início de cada *sprint* realiza-se uma reunião inicial (com duração de cerca de 1h), de forma a selecionar os requisitos do *Product Backlog* a implementar no *Sprint*. Uma vez iniciado o *Sprint*, acontece uma reunião diária (não mais de 15min) onde cada membro da equipa contribui com informação sobre o que fez, o que vai fazer e que obstáculos encontrou.

O objetivo é no final de cada iteração obter uma versão incremental do nosso produto, com os requisitos programados para esse sprint a serem satisfeitos e aceites pelo responsável pelo *Product Backlog*.

2.1.2 DevOps

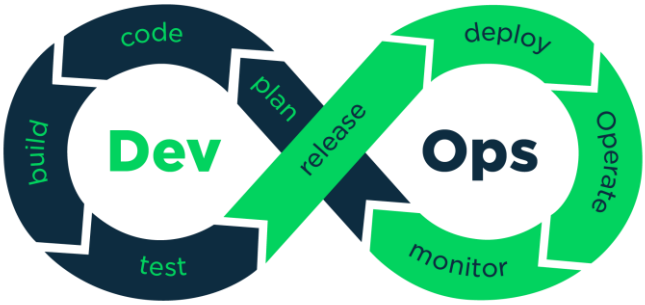

Figura 2.3 - DevOps lifecycle

A metodologia Agile permitiu que se seguisse o pipeline de DevOps[1]. DevOps consiste na aproximação entre equipas de *developers* e equipas de *operations* de modo a garantir que existe comunicação e colaboração por parte de ambos, originando ultimamente a realização de mais *releases*, com maior qualidade e maior estabilidade. Combinando com monitorização / manutenção constante, obtém-se assim uma taxa de falhas bastante menor, sendo essas falhas mais facilmente corrigidas.

DevOps não se foca em apenas entregar um produto com requisitos cumpridos, mas sim entregar um produto com valor alto associado, de forma contínua, permitindo assim, em projetos de larga escala, diminuir consideravelmente o esforço associado à gestão e entrega do mesmo.

2.2 Portable Class Library

Portable Class Library (*PCL*) é uma *library* portátil (vulgo *.dll*), tornando-se uma ótima estratégia para o aceleramento do desenvolvimento e para o aumento da reutilização do código. Permite a partilha de código juntando a lógica de uma aplicação num único componente, e definindo projetos diferentes para cada sistema operativo. Ao contrário do *Shared Project*, cujos conteúdos são diretamente copiados para cada tipo de sistema operativo, um projeto em PCL, após compilado, gera uma *library* que pode facilmente ser incluída nos outros projetos.

[1] DevOps é o termo usado para descrever um *pipeline* de planeamento, criação, teste, compilação e monitorização de uma aplicação.

2.3 Model-View-ViewModel

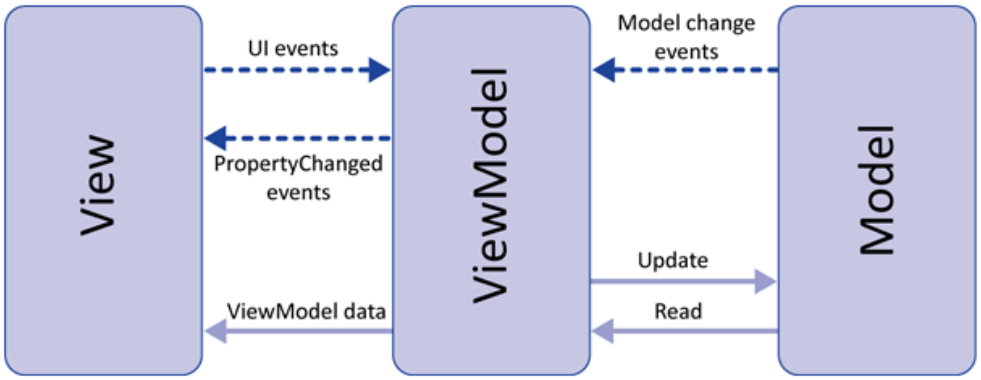

Figura 2.4 - Model-View-ViewModel [6]

Model-View-ViewModel (MVVM) é um padrão de arquitetura de software criado por *John Gossman*, focado em facilitar a separação do GUI do *back-end* [7] (Figura 2.4). Ao invés de utilizar um controlador para passar informação às *Views*, como no *Model-View-Controller* (MVC), o MVVM utiliza *ViewModels* que, por sua vez, usam *binders* para expor propriedades e comandos utilizados. Os *ViewModels* funcionam como mediadores da comunicação entre as *Views* e os *binders*, podendo converter cada valor em algo apresentável para o utilizador.

2.3.1 Binders

Os *Binders* são o *core* do MVVM. São os responsáveis pela separação do *layer* do desenvolvimento, do resto do padrão. [8] Ao criar esta separação as *Views* nunca estão dependentes dos objetos que lhe são passados, sendo que caso existam alterações frequentes a nível de interface numa fase mais avançada do projeto não será necessário alterar a lógica de negócio envolvida.

2.3.2 Dependency Injection

Dependency Injection [9] (DI) é um padrão bastante usado em MVVM que consiste na passagem de variáveis de instância aquando da criação de um novo objeto. Ou seja, se para criarmos o objeto A necessitamos de um objeto B, ao invés de fazermos a criação do objeto B, este é passado quando efetuamos a criação do objeto A. Uma boa analogia pode consistir ao criar um carro, ao invés de criamos a rodas ou a bateria, recebemos a implementação de ambas. [10]

2.3.3 Service Location

Service Location [11, 12] é um padrão arquitetural que tem como objetivo encapsular o modo como obtemos um serviço. De forma mais simples, permite a passagem de interfaces ao invés de objetos, de modo a que todas as chamadas que sejam efetuadas usando a interface, sejam redirecionadas para a respetiva implementação.

Na Figura 2.5 de *Martin Fowler* [13] conseguimos observar tanto *Service Location* como *Dependency Injection*:

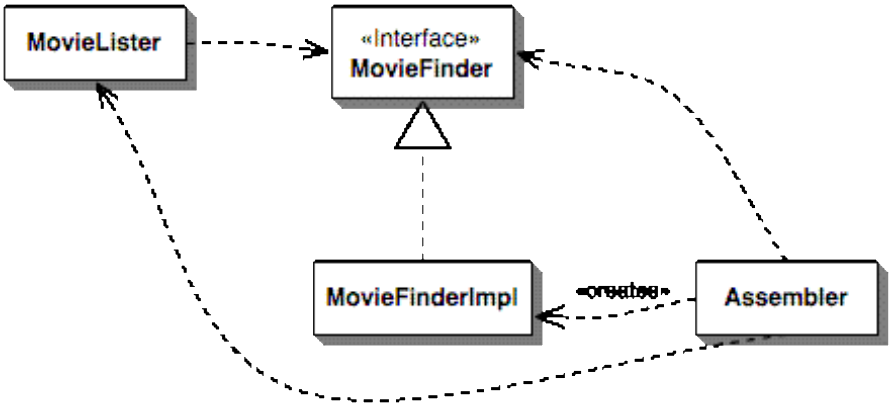

Figura 2.5 - Service Location and Dependency Injection

O Assembler necessita de criar um objeto do tipo *MovieLister*. Para tal passa-lhe a interface *MovieFinder* e trata de injetar uma implementação, neste caso o *MovieFinderImpl*.

2.3.4 Inversion of Control Container

Inversion of Control [14, 15, 16] (IoC) *container* é um objeto responsável pela gestão dos componentes de uma aplicação. Serve para simplificar a criação de novos objetos, fazendo uso de *Dependency Injection* e de *Service Location*. Na tradicional programação procedimental são pedaços de código específicos que fazem chamadas a *frameworks*. No IoC são as próprias *frameworks* que chamam código específico.

2.4 Platform as a Service

Platform as a Service [17, 18, 19] (PaaS) é um serviço de *cloud computing* que oferece uma plataforma para clientes desenvolverem e gerirem as suas aplicações. Os prestadores de PaaS conferem os servidores, bases de dados e outros serviços necessário para hospedar a aplicação, sendo já concebido para suportar todo o ciclo de vida de

aplicações. O utilizador gere as aplicações e os serviços que desenvolve e o fornecedor do serviço em *cloud* gere, tipicamente, tudo o resto [19]. Um exemplo de um prestador de PaaS é o Microsoft Azure[2], falado nos próximos capítulos.

[2] https://azure.microsoft.com/pt-pt/

Capítulo 3

Tecnologias utilizadas

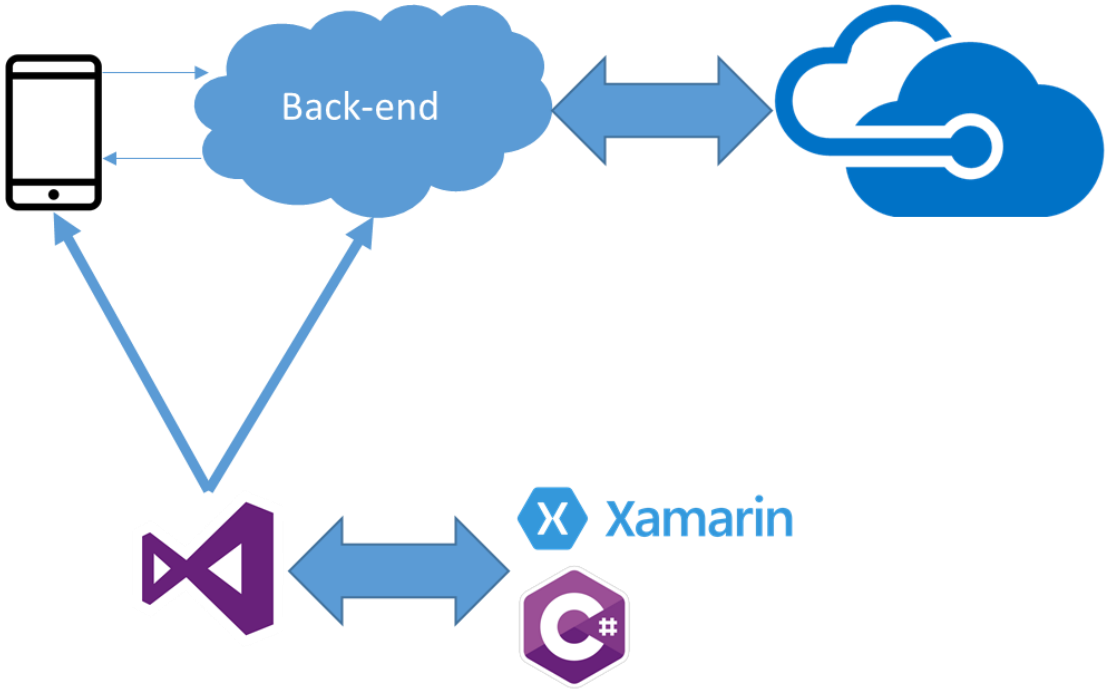

Figura 3.1 - Interligação entre as tecnologias utilizadas [20, 21, 22, 23]

Neste capítulo são apresentadas as tecnologias e a linguagem de programação utilizada durante o desenvolvimento da aplicação.

3.1 Microsoft *Visual Studio*

O Integrated Development Environment (IDE) utilizado neste projeto foi o Microsoft *Visual Studio* [24]. Este IDE suporta diversas linguagens de programação (C, C#, C++, Visual Basic, F#, etc). Atualmente está disponível para Windows com diversas versões estáveis, e com uma versão *preview* para Mac (lançada no final de novembro de 2016).

3.2 Xamarin

Atualmente existem inúmeros SO presentes nos *smartphones*, sendo Android, iOS e *Windows Phone* os mais utilizadoss (~86.8% de utilizadores em Android, ~12.5% em iOS e ~0.3% em *Windows Phone*) [25]. O paradigma de criação de uma aplicação começa

com cada SO necessitar de um IDE diferente que utiliza uma linguagem de programação diferente (Figura 3.2). O *Xamarin Platform* é uma das alternativas a essa abordagem.

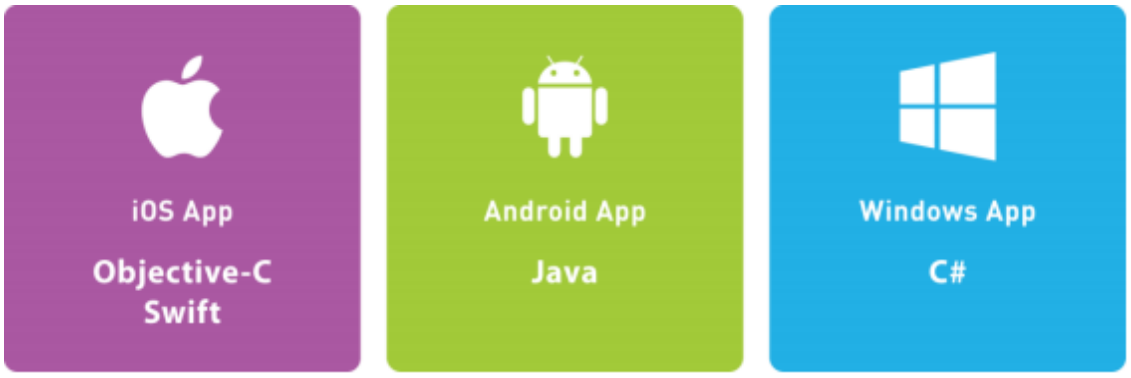

Figura 3.2 - SO's e respetivas linguagens de programação

O *Xamarin* é uma plataforma de desenvolvimento de aplicações para iOS, Android e *Windows Phone* (tal como de Windows e de Mac OS e até *WatchOS* e Android *Wear*) criada pela Xamarin.

Recorrendo a C# e ao *Visual Studio* (antigamente também ao *Xamarin Studio*, no entanto está em processo a sua descontinuação, com a introdução do *Visual Studio* para Mac), o *Xamarin* permite o desenvolvimento de aplicações *cross-platform* nativas.

3.3 C#

C# é uma linguagem de programação orientada a objetos, baseada em C++, Java e Pascal [26, 27]. Criada pela Microsoft, é a linguagem de programação usada pelo Xamarin. C# oferece uma imensidão de funcionalidades desde leitura e escrita de ficheiros, programação assíncrona (conceito importante em mobile, pois desta forma a aplicação não fica "bloqueada" à espera de resposta) e até *parse* de XML.

3.4 Azure

Azure é um conjunto de serviços integrados na *cloud* [28]. De entre os inúmeros serviços que disponibiliza, destacam-se neste projeto:

- Azure Mobile App.
- Azure SQL.
- *Application Insights*.
- Mobile Center.
- *Notification Hubs*.

- Azure *Search*.

Produto	Descrição
 **Mobile App Service**	A App móvel utiliza um *sdk* para se conectar a um *back-end* suportado sobre um serviço Azure Mobile App. Este SDK está preparado para suportar funcionalidades de autenticação e até suporte a funcionalidades offline.
 **SQL Database**	Informação estruturada é guardada num repositório de dados SQL, uma base de dados relacional sobre um serviço cloud, que suporta transações e *queries* eficientes.
 **Application Insights**	Este serviço tem a capacidade de detetar problemas, diagnosticar crashes e registar a utilização do *Mobile App Service*, suportando a tomada de decisão informada durante o ciclo de desenvolvimento e manutenção da solução.
 **VS Mobile Center**	Este componente substitui o produto *HockeyApp*, que permite recolher relatórios de crash e registar utilização da App móvel. Permite também a distribuição interna de versões das Apps.
 **Notification Hub**	Este serviço é utilizado para notificações *push* de forma escalável e *cross-platform*.
 **Azure Search**	Este componente permite efetuar uma pesquisa num índice anteriormente definido de modo a que o retorno dos resultados seja quase imediato.

Tabela 3.1 - Descrição dos serviços de Azure

3.5 Balsamiq

Balsamiq é uma aplicação usada para efetuar prototipagem da interface gráfica de forma a que o cliente possa aprovar o layout, antes de ele ser implementado. Facilita imenso cada iteração de uma funcionalidade pois oferece uma forma simples e rápida de gerar um *mockup* do ecrã em questão (Figura 3.3).

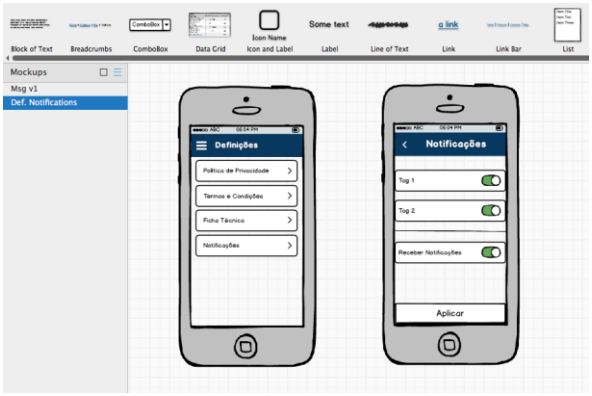

Figura 3.3 - Balsamiq

3.6 InvisionApp

O InvisionApp, é uma aplicação Web que corre no browser. Permite definir ecrãs (Figura 3.4) e algumas ações limitadas de navegação entre eles, de forma a "simular" a utilização de uma aplicação de telefone (Figura 3.5). Permitiu ilustrar várias ideias e sugestões, mostrando ao cliente a visão futura da aplicação, tal como possibilitou discussões sobre novas funcionalidades.

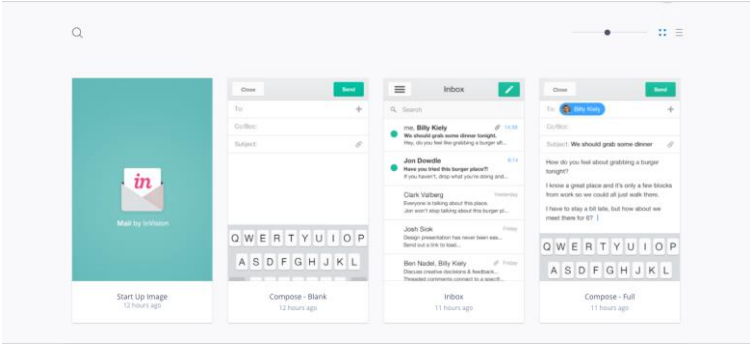

Figura 3.4 - Overview InvisionApp

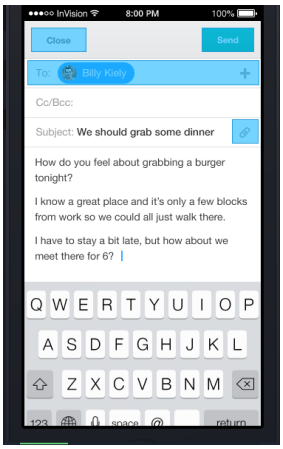

Figura 3.5 - Zonas de interação InvisionApp

Capítulo 4

Enquadramento

Neste capítulo é descrito o trabalho de investigação e pesquisa efetuado nomeadamente sobre *framework* alternativas, é dado enquadramento do projeto bem como a arquitetura da solução.

4.1 Xamarin.Forms

Xamarin.Forms parte do mesmo conceito que o Xamarin, com a diferença de efetuar também a partilha do código de UI. Ou seja, permite que a criação do UI seja feita apenas num local (Figura 4.1). A Microsoft aposta cada vez mais nesta abordagem pois permite o desenvolvimento de mais aplicações para *Windows Phone*[3], visto apenas ser necessário programar uma pequena parte de comportamento nativo, como a utilização de controlos de música por exemplo.

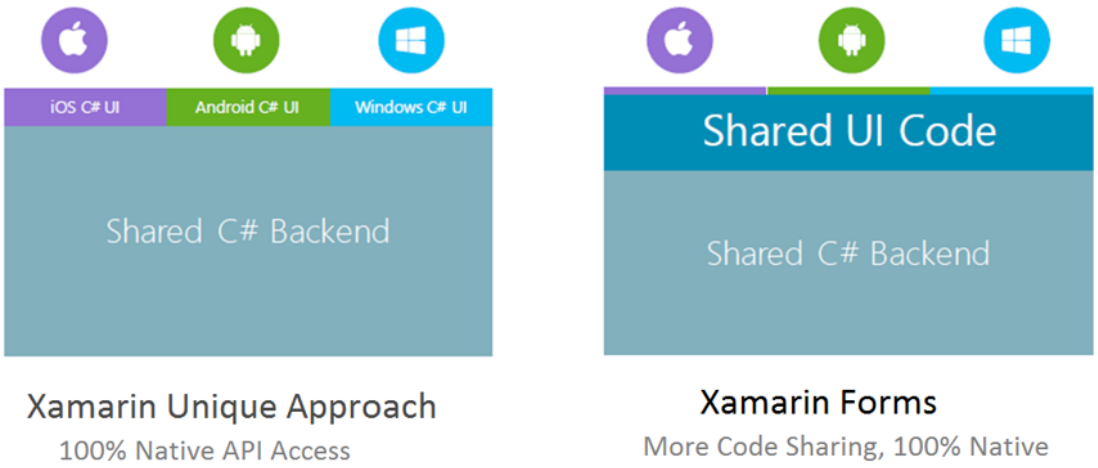

Figura 4.1 – Xamarin native vs Xamarin.Forms [29]

O problema que é muitas vezes associado ao Xamarin.Forms é o da performance da aplicação final. O Xamarin.Forms peca por não atingir o nível da performance do Xamarin nativo, o que em muitos dos casos torna-se um fator determinante na escolha da *framework*. No entanto, com a *preview* do Xamarin.Forms 3.0, o foco encontra-se na

[3] Neste caso são desenvolvidas para Universal Windows Platform (UWP), que permite que as aplicações possam correr também no desktop.

estabilidade e na melhoria da performance, tornando-se assim, de novo, numa excelente abordagem a seguir seja para o desenvolvimento de protótipos / provas de conceito, seja para o desenvolvimento de aplicações de produção.

4.2 React Native

React Native [30] é uma *framework open-source* desenvolvida pelo Facebook que faz uso de JavaScript para a criação de aplicações para Android e iOS. Permite a visualização de mudanças no código automaticamente, a integração de componentes nativos e a atualização da própria App sem ser necessário a passagem por nenhuma das *stores*. O problema atualmente associado ao *React Native* consiste na abordagem de lançamento de novas *releases*: o Facebook compromete-se a efetuar *releases* a cada duas semanas, normalmente com imensas correções de *bugs*, mas englobando também mudanças radicais, originando assim que muitos dos componentes usados possam ficar desatualizados.

4.3 OutSystems

OutSystems [31] consiste numa *framework* que prima por permitir aos *developers* criar rapidamente aplicações e serviços de *back-end* de suporte apenas arrastando componentes, sejam eles UI, modelos de dados ou até lógica de negócio, podendo fazer uso de qualquer funcionalidade nativa como sensores, obtendo uma performance um pouco melhor que a de uma aplicação híbrida[4]. Permite ainda efetuar o *deployment* de aplicações de forma simples, reverter caso ocorram erros e tirar métricas constantemente.

4.4 Comparação

	Xamarin nativo	Xamarin.Forms	React Native	OutSystems
Linguagem de Programação	C#	C#	JavaScript	Outsystems Markup Language

[4] Uma aplicação híbrida consiste numa aplicação que faz uso de componentes web como HTML, JavaScript e CSS, correndo num componente nativo, por exemplo uma WebView

Performance	Nativa	Equivalente a nativa	Equivalente a nativa	Próxima de Nativa
Facilidade de Customização	Alta	Média	Média	Alta
Facilidade de Desenvolvimento	Média	Alta	Média	Alta
Rapidez de desenvolvimento	Média	Rápida	Média	Rápida
Facilidade de Deployment	Média-Baixa	Média-Baixa	Média	Simples
Plataformas Suportadas	Android, iOS, UWP	Android, iOS, UWP	Android, iOS	Android, iOS, Web

Tabela 4.1 - Comparação entre frameworks para Mobile

Observando a tabela acima, facilmente podemos observar prós e contras para cada uma das *frameworks*. Xamarin e Xamarin.Forms consistem ambos numa boa abordagem para o desenvolvimento de aplicações de produção, sendo que caso a aplicação consista em apresentação de formulários e não necessite de mudanças radicais de layout entre plataformas, Forms é a melhor opção. *React Native* é indicado caso a equipa de desenvolvimento tenha bases em JavaScript. Finalmente *OutSystems* oferece a possibilidade de construção rápida de uma aplicação com uma UI polida, com a troca de uma pequena perca de performance. De notar que a escolha da utilização de Xamarin nativo para o projeto ficou nas mãos da empresa.

4.5 Padrão de navegação

Um dos primeiros desafios recaiu sobre o padrão de navegação que iria ser usado. O padrão nativo do Android passa pelo uso de um *Hamburger menu* (ou *Naviga*tion *Drawer*) que possui todas as funcionalidades existentes na aplicação, enquanto que no iOS o mais comum é o uso de *tabs* que se encontram no fundo do ecrã.

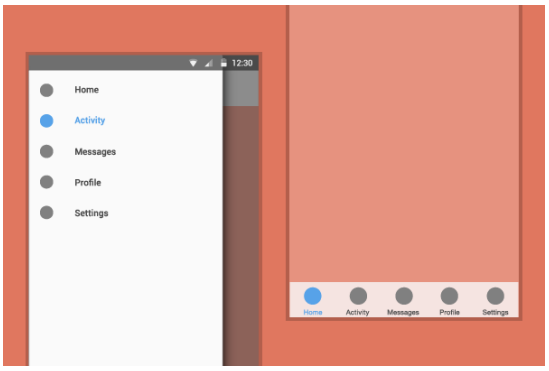

Figura 4.2 - Diferenças de navegação (Android esquerda, iOS direita) [32]

Durante as primeiras reuniões com o cliente, foi notório o imenso número de funcionalidades que planeava incluir na aplicação, fazendo com que o uso de *tabs* em iOS se tornasse inexequível. Iria-se então implementar um menu também em iOS, mesmo não sendo um comportamento nativo. Esta abordagem originou algumas diferenças de design para o Android, por exemplo no modo de abertura do menu, no entanto a funcionalidade manteve-se inalterada. Este princípio de navegação permite, desde que numa página de 1º nível, aceder a qualquer outra funcionalidade presente na aplicação, como podemos observar na Figura 4.3.

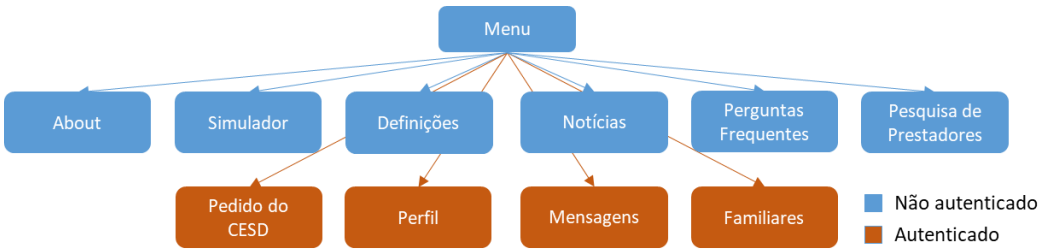

Figura 4.3 - Modo de navegação

4.6 Desenho da Solução

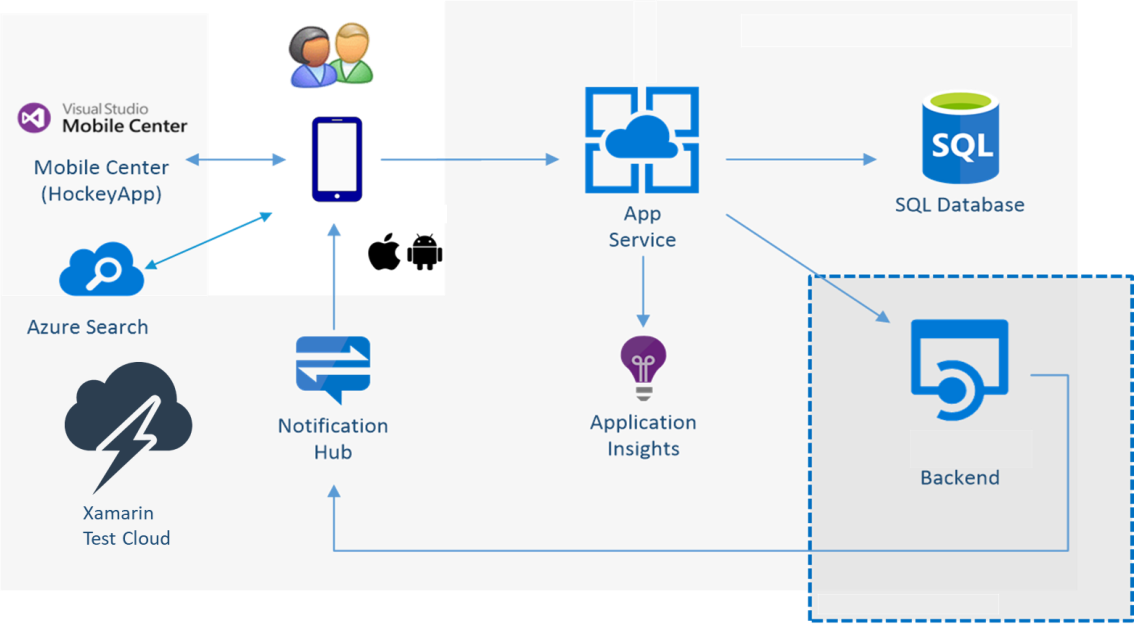

Figura 4.4 - Arquitetura solução

Foi acordado que na fase inicial do projeto cada programador iria focar-se apenas num sistema operativo. Desta forma, irá ser descrita apenas a implementação das *Views* em Android, tal como o código que ambos os sistemas operativos partilham.

A aplicação foi desenvolvida em PCL, dividindo-se no *Core*, no *Droid* (*Views* de Android) e no *Touch* (*Views* de iOS). Como se usou MVVM, toda a lógica da aplicação situou-se no *Core* enquanto que toda a camada de apresentação se encontra nos respetivos projetos. Para além dos três projetos, foram também criados projetos de teste de modo a assegurar o funcionamento esperado e equiparado entre sistemas operativos.

Como descrito anteriormente, o *back-end* é composto por um PaaS em Azure, que posteriormente acede aos serviços disponibilizados pelo cliente. A aplicação efetua pedidos ao *Mobile App Service* que pode ou não aceder aos serviços de *back-end* do cliente e à base de dados presente no Azure, sendo que em todas as chamadas são feitos diversos *logs* para o *Application Insights*. Antes e depois da receção da resposta por parte do *Mobile App Service*, são efetuados *logs* para o *Mobile Center* de modo a possibilitar o registo de métricas do lado da app. A aplicação pode também efetuar pedidos diretamente ao Azure *Search*, sem ser necessário passar pelo *Mobile App Service*, sendo, no entanto, também registados no *Mobile Center*.

O *Mobile App Service* efetua pedidos à maquina virtual presente no Azure do cliente, que por sua vez realiza os pedidos à máquina virtual física nas instalações do

cliente. Tanto a ligação entre o *Mobile App Service* e a máquina virtual presente no Azure do cliente, tal como a ligação entre ambas as máquinas do cliente estão protegidas por *Network Security Groups*[5] (NSGs) [33]. Qualquer chamada da aplicação ao *back-end* irá sempre ser dirigida ao *Mobile App Service*, sendo que pode receber os dados diretamente do serviço / base de dados presente no Azure, ou então do serviço de *back-end* do cliente.

[5] NSGs funcionam como *firewalls* para *virtual networks*. Permitem aplicar regras de tráfego a todos os recursos na mesma *virtual network*.

Capítulo 5

Trabalho Realizado – App

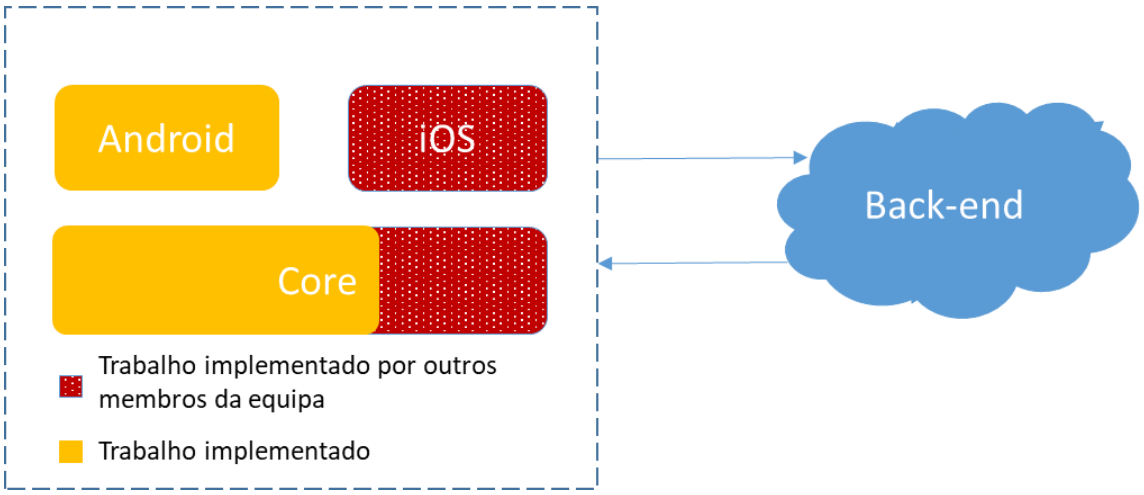

Figura 5.1 - Trabalho realizado do lado da App

Como foi usado a *framework* MVVM, fez-se uso de *ViewModels* para ligar toda a lógica às *Views*, que, desta forma, seriam alheias ao modelo de dados utilizado, assegurando assim que o *Front-End* e *Back-End* da app seriam autónomos entre si.

A estrutura do projeto está dividida entre quatro principais secções: *Models*¸ *Services*, e *ViewModels*. Os *Models* representam os modelos de dados usados na aplicação; os *Services* contêm todas as classes que fazem pedidos ao *back-end*. Todos os serviços são inicializados como *Singletons*[6] e é o IoC *container* que trata da sua criação automaticamente, fazendo uso de *Dependency Injection* e *Service Location*, como falado anteriormente; as Interfaces, como o nome indica, contêm todas as interfaces usadas pelo IoC para inicialização dos *Services;* finalmente os *ViewModels* efetuam a ligação da lógica às Views, tal como ilustrado na Figura 5.2, fazendo uso das Interfaces para inicialização (1) e dos *Services* (2) para executar chamadas ao *back-end* (3) de modo a retornar objetos usando os modelos de dados presentes nos *Models* (4) de volta para o *ViewModel* (5).

[6] Um *singleton* corresponde a uma classe apenas poder ter uma instância de si própria criada na duração da aplicação, ou seja, sempre que essa classe é referenciada, primeiro é procurada uma instância dessa mesma classe. Caso exista, é usada essa instância, caso não exista, então é criada uma nova e é usada sempre essa [22].

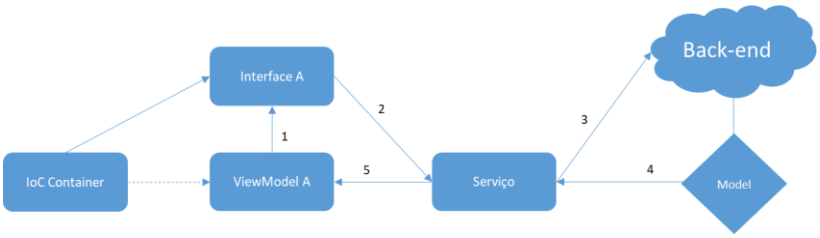

Figura 5.2 - Diagrama de uma chamada ao back-end

Neste capítulo são detalhadas as funcionalidades implementadas do lado da App, bem como todos os ecrãs disponíveis por opção de menu. No final são brevemente referidos os documentos de suporte criados paralelamente à implementação.

5.1 Offline Sync

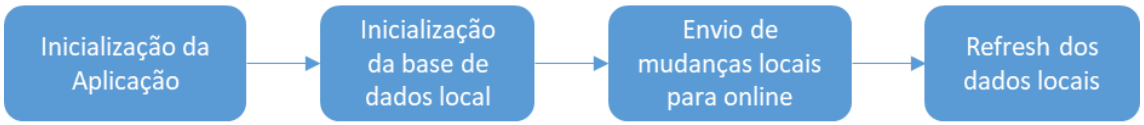

Figura 5.3 - Diagrama de funcionamento do Offline Sync

Após algumas reuniões como o cliente e após as propostas de implementação de algumas funcionalidades terem sido aceites, tornou-se claro de que alguns dos conteúdos poderiam ser disponibilizados em modo offline, como por exemplo conteúdos de ajuda. Para tal, implementou-se na App a funcionalidade de *Offline Sync*.

O *Offline Sync* é uma característica do Azure *Mobile Apps* em que é permitido ao utilizador efetuar interações com elementos de uma base de dados, até quando não existe ligação à internet. Qualquer mudança é guardada numa base de dados local, e posteriormente sincronizada quando a ligação é reposta. Neste caso apenas é permitido ao utilizador efetuar ações de consulta a registos da base de dados, não sendo permitidas alterações. O processo de implementação do *Offline Sync* necessita de apenas 3 passos:

1. Criação do *Mobile Service Client* que fará o acesso ao *back-end*.
2. Criação das tabelas locais para posterior sincronização com as tabelas do Azure SQL.
3. Sincronização de alterações com as tabelas do Azure SQL.

De notar que no terceiro passo do diagrama da Figura 5.3, quaisquer alterações locais são enviadas primeiro para a base de dados online, onde são *merged*[7] fazendo uso

[7] *Merge* é o processo no qual informação que foi alterada em dois locais é junta de forma a que o resultado seja contenha ambas as mudanças.

de campos específicos explicados do lado do *back-end*, e só depois é feito o pedido para atualização dos conteúdos locais.

5.2 Azure Search

O Azure *Search* é um serviço de pesquisa na *cloud* onde o servidor e a infraestrutura são geridos pela Microsoft, havendo a possibilidade de replicar ou adicionar partições de modo a que a performance de pesquisa não se degrade com a diferença de localização ou aumento de pedidos. Com alta disponibilidade, permite a integração com serviços de armazenamento Azure, como Azure SQL ou *DocumentDB*, enquanto coleciona informação sobre tráfego e oferece a possibilidade de criação *queries* a partir do portal do Azure.

O Azure *Search* possui um SDK para .NET, sendo que para integrar numa aplicação apenas é necessário proceder à criação de um *Index Client*, que fará ligação ao serviço presente no *back-end*, definir os parâmetros de pesquisa e efetuar a query. Alguns dos parâmetros de pesquisa disponíveis são:

- *Select*: retorna apenas as colunas selecionadas.
- *Query Type*: permite que a pesquisa use a sintaxe *Lucene. Lucene* permite que a pesquisa por texto seja bastante mais completa do que uma pesquisa normal. Permite a utilização de operadores *Boolean*, de *Fuzzy Search*[8], *Term Boosting*[9], etc.
- *Search Mode*: determina que algum ou todos os termos de pesquisa têm de coincidir.
- *Top*: determina o número máximo de resultados a devolver.

De momento para além de *Fuzzy Search*, é também usada a predição. Ou seja, caso o utilizador apenas escreva uma ou duas letras, o Azure *Search* calcula as possíveis palavras, faz a pesquisa com esse cálculo e retorna os resultados relativos a uma pesquisa usando essas palavras. Após alguns testes foi notória o aumento de performance do Azure *Search* em relação ao mecanismo de pesquisa já existente do cliente, sendo os resultados

[8] *Fuzzy Search* é um modo de pesquisa onde não pesquisa apenas por um termo, mas por uma aproximação desse termo. Por exemplo, uma *fuzzy search* de um nível na palavra "*blue*" pesquisaria por "*blue*", "*blues*", "*glue*", etc.

[9] *Term Boosting* é usado para atribuir uma pontuação de pesquisa mais elevada a termos previamente definidos.

não só mais relevantes como também a resposta é bastante mais rápida tanto em pesquisas simples como complexas.

5.3 Certificate Pinning

De modo a prevenir ataques de *Man-In-The-Middle* (MitM)[10], todas as comunicações retornadas à aplicação são testadas de forma a garantir que provêm de fonte segura, neste caso do *Mobile App Service* ou do Azure *Search*. Para tal, após a receção, é extraída a chave pública do certificado de onde foi enviada a informação, e apenas caso exista na lista de chaves aceites é processada.

5.4 Autenticação

5.4.1 Login

O Login é efetuado fazendo uso ao *sdk* do Azure *Mobile App*, que por sua vez faz uma chamada ao *Mobile App Service* que trata da validação das credenciais. O resultado, se as credenciais estiverem corretas, vem sob a forma de um *Json Web Token* (JWT) e do *userid* associado ao utilizador.

5.4.2 Segurança

O processo de Login é efetuado em dois momentos, com dois passos em cada momento:

- Observando a Figura 5.4, quando o utilizador liga a aplicação, esta verifica se tem algum *token* guardado nas definições da aplicação. Caso tenha, então procede à validação desse mesmo *token*. Em caso de sucesso procura ir buscar as informações sobre esse mesmo utilizador, deste modo garantindo que caso tenha havido uma atualização dos dados, ter-se-á sempre os mesmos corretos.

[10] *Man-In-The-Middle* é um ataque onde as comunicações entre duas entidades são intercetadas sem que os utilizadores se apercebam que não estão a comunicar com quem pensam, considerando assim que os dados recebidos são válidos.

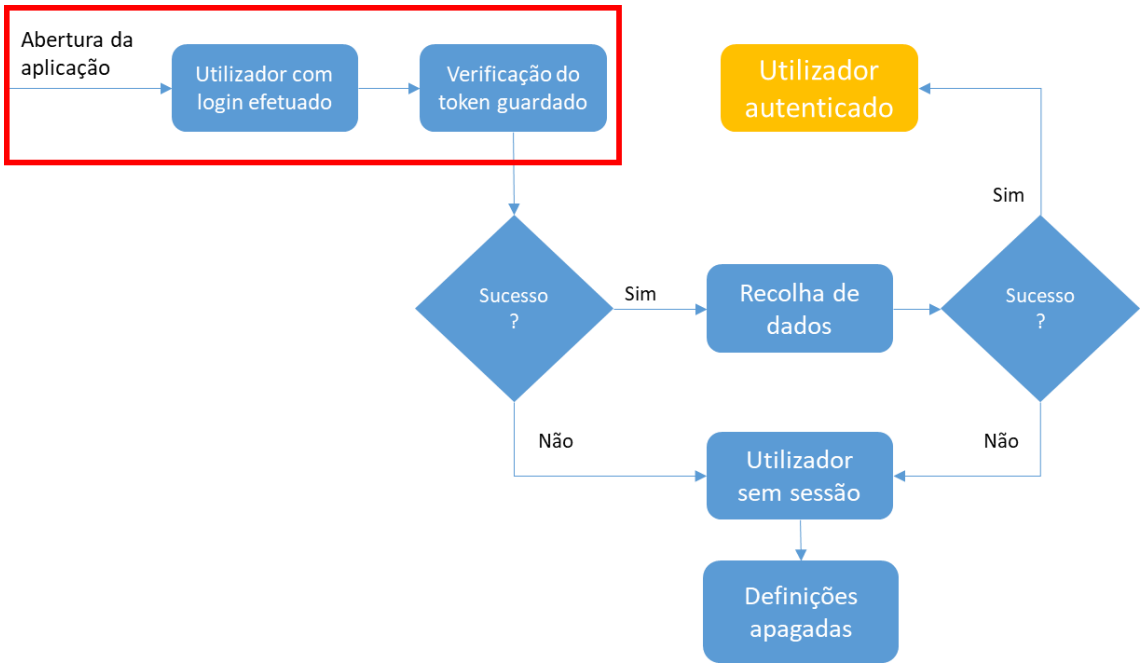

Figura 5.4 - Fluxo da chamada de verificação do Token

- Observando a Figura 5.5, quando o utilizador navega até ao ecrã de login e tenta fazer o login usando as suas credenciais, é primeiro testado se as credenciais estão corretas. Caso estejam, então é efetuada outra chamada para ir buscar as informações desse utilizador.

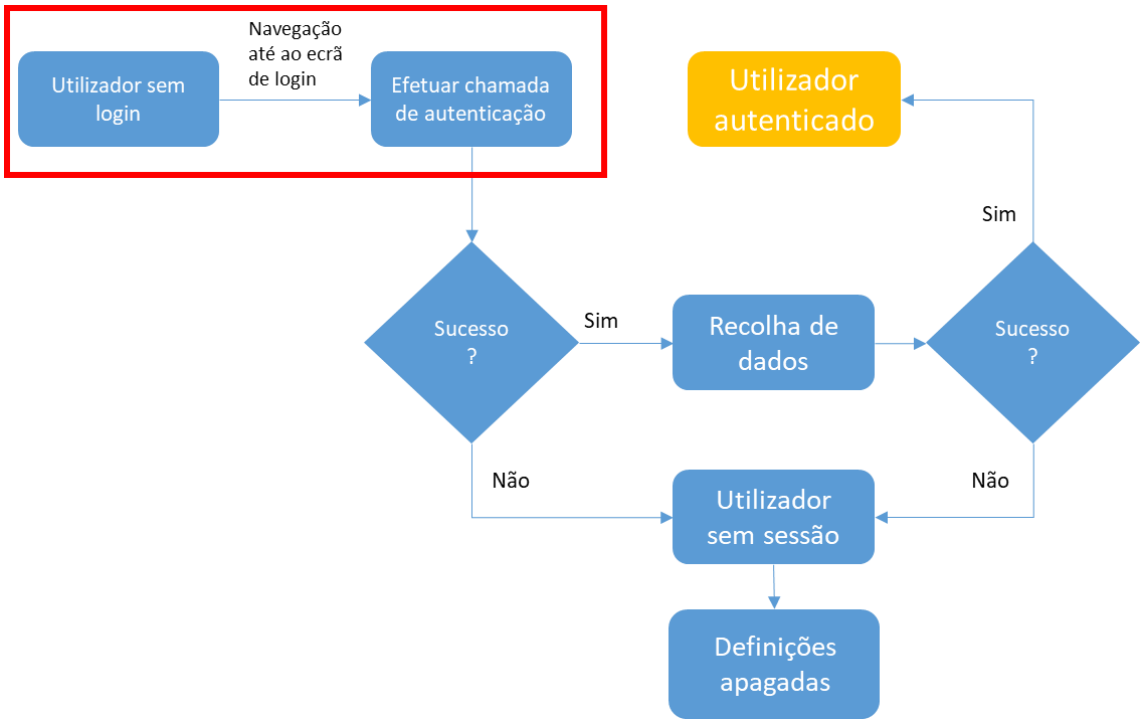

Figura 5.5 - Fluxo da chamada de verificação das credenciais

Na eventualidade de algum destes passos falhar, então todas as definições da app são apagadas e a aplicação é iniciada em modo não-autenticado. De notar que qualquer comunicação é feita de forma segura por HTTPS.

5.5 Notícias

As notícias são o primeiro ecrã mostrado ao utilizador, funcionando como *homepage*. É composto por uma lista de *scroll* horizontal reservada para as notícias em destaque, seguido de uma lista de *scroll* vertical para as restantes notícias. De notar que uma notícia poder ser simultaneamente *highlight* e normal, sendo que a aplicação faz uso de diferentes imagens em ambos os casos, estando essas imagens presentes do lado do cliente. A pedido do cliente as notícias são disponibilizadas em modo offline, fazendo uso do *Offline Sync* referido anteriormente.

5.6 Perfil

O Perfil é representado por listas em que cada campo contém uma chave e um valor. De modo a ser mais fácil a mudança dos dados a apresentar, a aplicação não controla quais os pares de valores que lhe são passados, ficando ao critério do *back-end* do cliente. Existem diferentes tipos de utilizador, cada um com apresentações diferentes no ecrã do Perfil:

1. O utilizador é titular, e não possui representante, tendo neste caso apenas duas *tabs*: dados pessoais e contactos.
2. O utilizador é titular, e possui representante, tendo neste caso quatro *tabs*, duas do utilizador, dados pessoais e contactos, e as restantes duas do representante, dados pessoais e contactos.
3. O utilizador não é titular e possui representante, tendo assim três *tabs*: dados pessoais do utilizador, dados pessoais e contactos do representante.

5.6.1 Edição de Perfil

Um dos problemas expostos pelo nosso cliente consistia nas diversas reclamações dos utilizadores relativamente a dados incorretos. Para tal foi disponibilizado sob a forma de um ecrã de entradas de texto, uma forma simples do utilizador poder editar os seus dados pessoais e os seus contactos (Figura 5.6). São apresentados ecrãs distintos

dependendo se o utilizador é ou não titular. Em ambos os casos, existe informação que não é possível editar, por exemplo o número de utente ou o NIF.

Figura 5.6 - Navegação para Edição de Perfil

5.6.2 Cartão Virtual

O Cartão Virtual é a verdadeira desmaterialização de um artefacto que até então era exclusivamente físico. Esta funcionalidade permite reduzir custos de emissão e reemissão tal como é a forma mais imediata da confirmação da validade dos direitos do utente. É composto por um único ecrã em *landscape* contendo alguma informação sobre o estado do utilizador, tal como a data de geração de modo a assegurar que o utilizador se encontra com direitos no mesmo dia que o cartão é mostrado. A data de geração é controlada do lado do *back-end* de modo a assegurar que mesmo alterando a data do dispositivo, aquela data não se altera.

5.7 About

Neste ecrã separado em *tabs* são apresentados diversos dados sobre o cliente, nomeadamente uma breve explicação de "Quem somos" sob a forma de uma *Webview*, uma secção de "Contactos" com opções de *ClickToAction* dependendo do tipo de contacto (Telefone, Email ou Website) e finalmente "Onde Estamos", onde é apresentado um mapa com marcadores das localizações dos postos de atendimento.

5.8 Perguntas Frequentes

O esquema das perguntas frequentes está organizado em Categorias e em Subcategorias, sendo ambas apresentadas em forma de lista. Após selecionar uma Categoria, e posteriormente uma Subcategoria, é exibido o conteúdo de ajuda sob forma

de uma *Webview*. Estando também disponível em modo offline, existe uma base de dados própria onde é possível guardar todos os conteúdos. Desta forma o cliente pode adicionar, remover ou atualizar toda a informação disponível desta funcionalidade.

5.9 Pesquisa de Prestadores de Serviço

Esta funcionalidade revelou-se como a mais complexa devido à lógica associada. Este ecrã permite ao utilizador efetuar uma pesquisa fazendo uso de três filtros (Figura 5.7):

- O quê: pesquisa por texto livre.
- Onde: pesquisa pela localização.
- Categorias: pesquisa por categorias específicas.

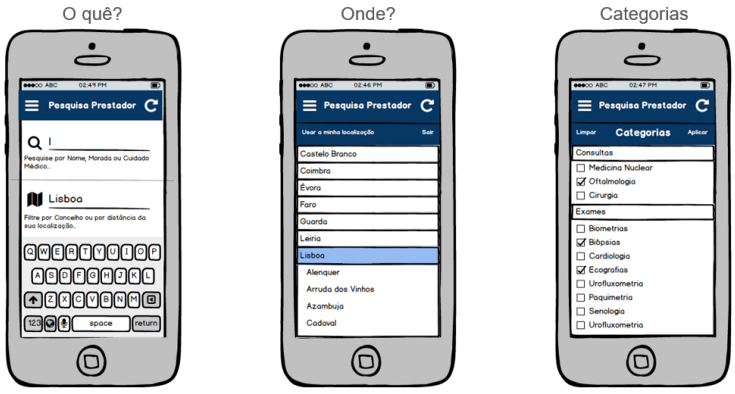

Figura 5.7 - Filtros da Pesquisa de Prestadores

A pesquisa pode englobar mais do que um filtro simultaneamente. Apesar de ser permitido a pesquisa por mais do que uma categoria, apenas é permitido a pesquisa por um único concelho. No ecrã de resultados é apresentada uma lista de cartões (Figura 5.8), onde estão apenas representados o nome, a morada e a cidade. No topo do ecrã é apresentada uma barra que detalha o número de resultados existentes para essa pesquisa. É também possível a visualização da lista de prestadores sobre a forma de mapa (Figura 5.9). Neste caso é apresentado um mapa com *pins* nos respetivos locais.

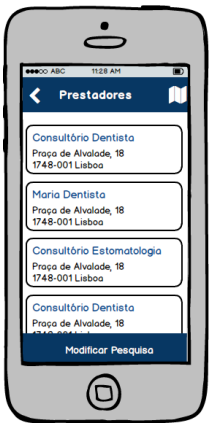

Figura 5.8 - Vista de Cartões

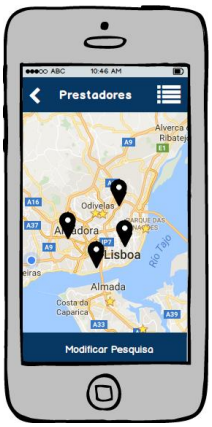

Figura 5.9 - Vista em Mapa

O utilizador ao escolher um dos resultados (seja pelo cartão na lista, ou pelo nome no *pin* no mapa), é levado para um ecrã de detalhe. Nesse ecrã é apresentada uma imagem da localização do prestador, informação de contacto do mesmo, tal como a lista de categorias associadas.

5.10 Simulador

O simulador permite a pesquisa por texto de todas as categorias existentes, fazendo uso de *fuzzy search* e pesquisa preditiva. Quando o ecrã é mostrado pela primeira vez, ou quando o utilizador não preenche o campo de pesquisa, são apresentadas as 20 categorias efetuadas mais vezes nos últimos 365 dias. Caso o utilizador insira uma pesquisa válida, são apresentados os resultados correspondentes, ordenados por nome (a pedido do cliente). Após a seleção do resultado pretendido, são mostrados dois campos para preenchimento: a quantidade, e o custo unitário. Caso o utilizador preencha corretamente ambos os campos e efetue a simulação, é então exibido o possível valor de reembolso associado. De notar que, em ambos esses ecrãs, são apresentadas as regras associadas a essa categoria.

5.11 Mensagens

As mensagens são uma forma de alerta para os diferentes eventos a que um utilizador está subscrito, tendo desta forma uma visão global das notificações de todos os eventos. O utilizador recebe também ao longo das primeiras utilizações da aplicação um guia semanal de modo a poder explorar todas as funcionalidades existentes. A abertura de cada mensagem despoleta um processo do lado do *back-end* de modo a poder marcar a mensagem como lida (Figura 5.10).

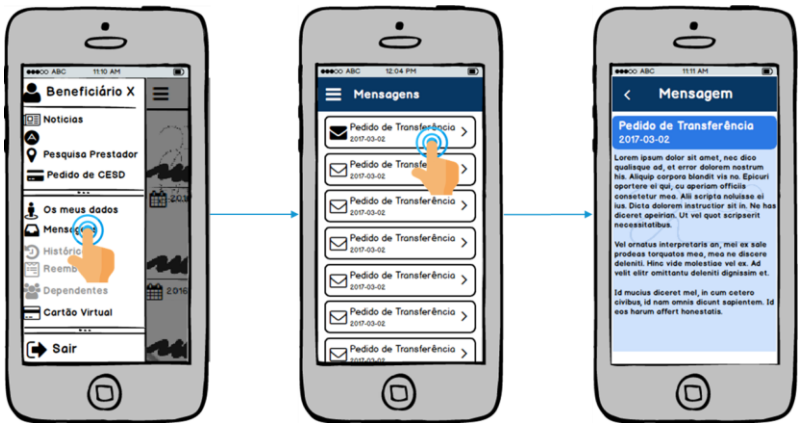

Figura 5.10 - Funcionalidade das Mensagens

5.12 Pedido do Cartão Europeu de Saúde e Doença

Esta funcionalidade permite ao utilizador efetuar o pedido do Cartão Europeu de Saúde e Doença (CESD) para si e/ou algum dos seus familiares. Consiste num layout simples com *checkboxes* para assinalar para quem deve ser pedido o CESD, seguido de quatro campos de preenchimento referentes ao nome da pessoa a contactar, morada de envio, email de contacto e comentários adicionais, sendo que os campos podem vir pré-preenchidos com a informação presente no Perfil, caso esteja disponível (Figura 5.11). Ao enviar é pedida confirmação antes de efetuar a chamada para o *Mobile App Service*, de modo a mitigar possíveis envios por engano.

Figura 5.11 - Abertura da funcionalidade do Pedido do CESD

Está também disponível uma opção de ajuda no canto superior direito que oferece informações sobre o CESD, tal como alguns links úteis para outro tipo de informações, nomeadamente informações de casos especiais (Figura 5.12).

Figura 5.12 - Ecrã de apoio ao Pedido do CESD

5.13 Documentação

5.13.1 Deployment da aplicação

De forma a que o *deployment* das aplicações ocorresse de modo quase automático, foram criados documentos de lançamento das aplicações nas respetivas lojas. Estes documentos detalham os diferentes passos necessários desde a criação da *Keystore*[11] para Android ou do *Provision Profile*[12] para iOS, até aos documentos de suporte necessários para finalizar ambos os lançamentos.

5.13.2 Geração de executáveis

Para um melhor entendimento do processo de *release* criou-se um documento detalhando o processo de criação de executáveis (.ipa, .apk) de modo a mitigar a ocorrência de imprevistos. Detalha como é feito a geração usando o *Visual Studio*, passando também pelos ficheiros necessários para assinar um executável e como é possível obtê-los.

5.13.3 Guidelines Xamarin

Procurando homogeneizar o código foi criado um documento de normas de programação, paralelamente ao desenvolvimento do projeto, exemplificando qual o

[11] *Keystore* é o ficheiro contendo uma ou mais chaves privadas. Ao assinar uma aplicação, o certificado da chave pública é anexado à aplicação. Desta forma, é possível garantir que futuros lançamentos e *updates* a uma aplicação são provenientes da mesma fonte.

[12] *Provision Profile* é o equivalente da combinação entre *keystore* e *manifest* para iOS que permite identificar unicamente uma aplicação a um distribuidor (utilizador / equipa de desenvolvimento / empresa), de modo a que qualquer futuro lançamento da mesma aplicação provenha do mesmo distribuidor.

standard para os métodos / classes / variáveis. Foi assim possível redigir código mais limpo e legível.

Capítulo 6

Trabalho Realizado - Back-end

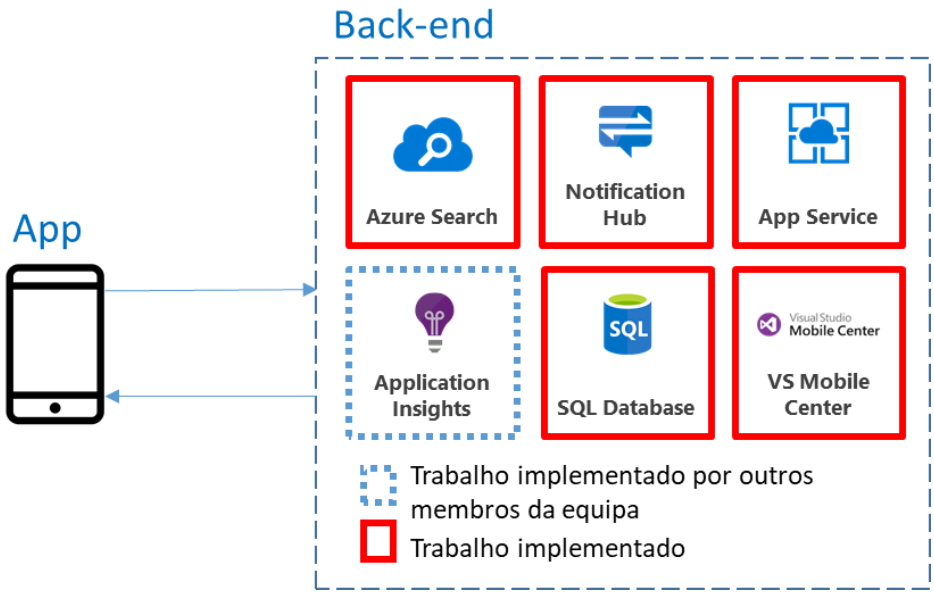

Figura 6.1 – Trabalho realizado do lado do back-end

O *back-end* funciona como serviço de apoio à App. Consiste num *webservice* que disponibiliza todas as chamadas fazendo uso de *ApiControllers*[13] para que o processamento não seja efetuado do lado da App. Através do *Mobile App Service* a App consegue efetuar autenticação, aceder à base de dados e realizar pedidos ao *back-end* do cliente. Paralelamente ao *App Service* estão presentes alguns serviços e funcionalidades como o *Notification Hubs* e o *Offline Sync* respetivamente, tal como o *Application Insights* que efetua a recolha de métricas e *logs*. Todos os componentes descritos estão representados na Figura 6.1.

6.1 Offline Sync

O *Offline Sync* do lado do *back-end* consiste na criação dos modelos de dados e das tabelas que serão disponibilizadas offline tal como dos *TableControllers* que as expõem.

[13] ApiController é um controlador para uma função que pode ser chamada do lado da App de modo a retornar dados sem ser necessário a App efetuar chamadas diretamente ao *back-end* do cliente ou à base de dados

6.1.1 TableController

Na criação do controlador, ao invés de ser um *ApiController* normal, usou-se um *TableController*. Isto porque como apenas desejamos expor a tabela e apenas métodos CRUD[14], a criação de um *TableController* no *Visual Studio* gera automaticamente esses métodos por nós.

6.1.2 Modelos de dados

```
public class FAQItem : EntityData
   {
       public string CategoryId { get; set; }

       [MaxLength(300)]
       public string Title { get; set; }

       public string Body { get; set; }

       public int Order { get; set; }

       public DateTime PublishedOn { get; set; }

       public bool isActive { get; set; }
   }
```

Figura 6.2 - EntityData Model

Todos os modelos de dados que se queiram disponibilizar *offline* deverão herdar da classe *EntityData (*Figura 6.2*)*. Tornar um modelo de dados num *EntityData* permite que o controlador que expõe este modelo consiga, fazendo uso de 5 campos, saber que alterações é que a base de dados local sofreu em relação à base de dados remota, sendo que esses campos são:

- *ID* - GUID único que identifica um registo.
- *Version* - Array de bytes que representa a versão de um registo.
- *CreatedAt* - Data de criação do registo.
- *UpdatedAt* - Data de atualização do registo. É usado para o *TableController* saber qual a versão mais recente de um determinado ficheiro.
- *Deleted* - *Boolean* que determina se um registo já foi apagado ou não. Permite assim rastreabilidade dos registos apagados.

[14] CRUD ou *Create*, *Read*, *Update* e *Delete* são as operações básicas de uma tabela presente na base de dados.

6.2 Xamarin Test Cloud

Testes são um componente integral de qualquer projeto, tanto que no tempo atribuído à implementação de uma funcionalidade é sempre alocado já com testes em mente. Existem diferentes tipos de teste, mas focou-se apenas em três: unitários, integração e UI.

- Testes unitários focam-se em garantir o correto comportamento de pequenas partes individuais do código.
- Testes de integração asseguram que a junção de diversos componentes funciona corretamente.
- Testes de UI como o nome indica recaem sobre a *User Interface*, de modo a garantir que a interface se mantém coerente nos vários dispositivos.

Os testes de UI podem ser criados de duas formas diferentes: ou por código, em que é estabelecido o fluxo de interação paralelamente à navegação manual no dispositivo, ou usando o *Xamarin Test Recorder*.

6.2.1 Xamarin Test Recorder

O *Xamarin Test Recorder* consiste num componente que permite ao utilizador, à medida que navega pela aplicação, gerar o código associado a essa navegação de modo a, ao executar esse mesmo código, efetuar o mesmo fluxo de ações. É constituído por uma interface simples (Figura 6.3) onde ao efetuar toques no dispositivo selecionado, o código é automaticamente gerado, podendo no final ser exportado diretamente para o *Visual Studio*.

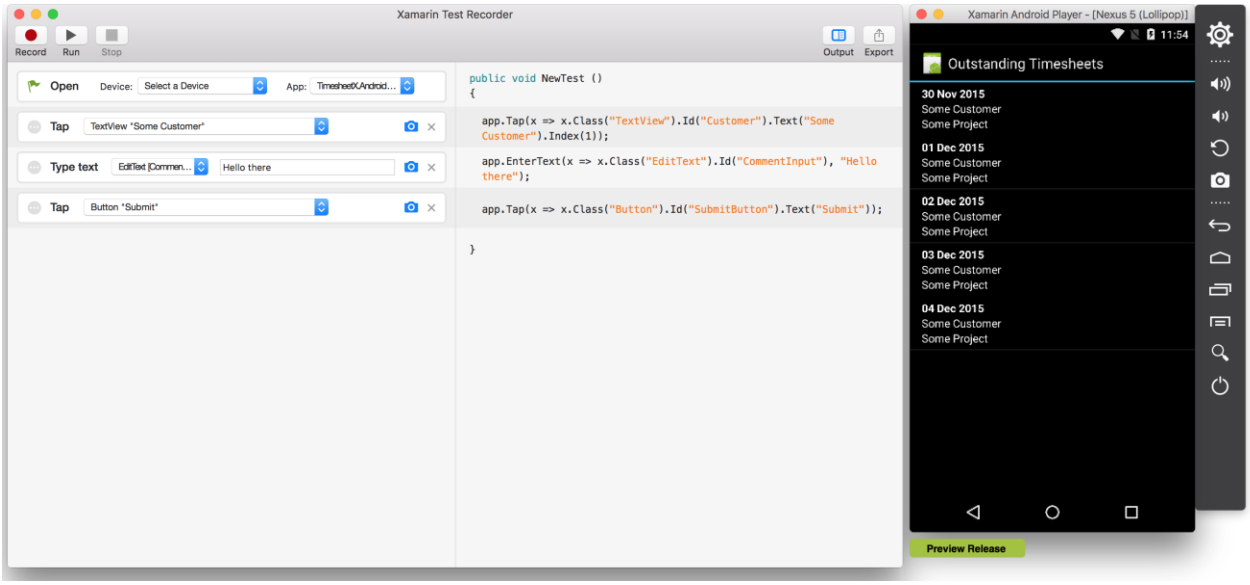

Figura 6.3 - Xamarin Test Recorder [34]

6.2.2 Utilização do Xamarin Test Cloud

Devido ao vasto leque de dispositivos existentes, especialmente em Android, torna-se difícil adquirir equipamentos com características diferentes. Surge assim a necessidade de responder de forma virtual a um problema que é, na verdade, físico, sendo para tal utilizada uma plataforma de testes - o *Xamarin Test Cloud*.

O *Test Cloud* permite, a partir do *Visual Studio*, fazer *upload* dos testes de UI diretamente de modo a que apenas seja necessário selecionar os dispositivos em que pretendemos executar de um conjunto de mais de 2700 dispositivos possíveis. Após correr os testes especificados apresenta um ecrã contendo a *overview* de todos os testes corridos numa dada aplicação (Figura 6.4), bem como alguns detalhes sobre cada bateria de testes corrida (Figura 6.5)

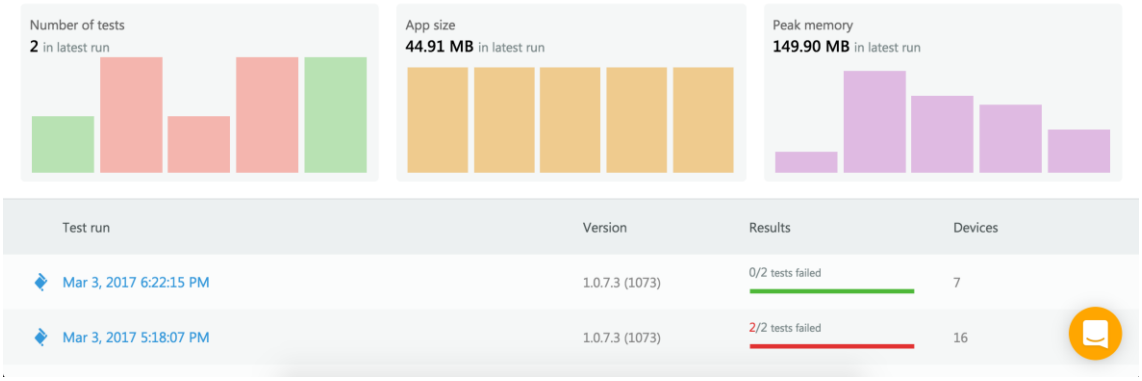

Figura 6.4 - Xamarin Test Cloud

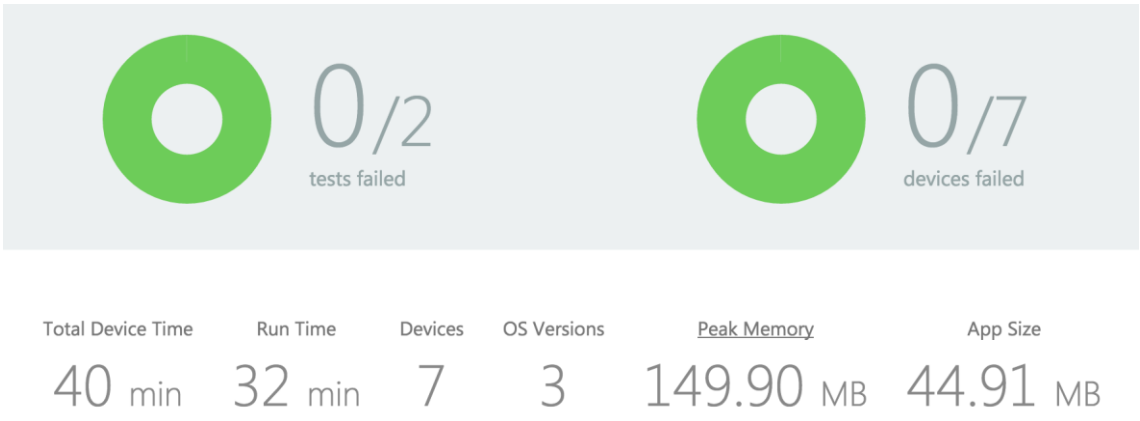

Figura 6.5 - Detalhes do teste

Para além de apresentar estatísticas sobre os testes, é também possível ter *screenshots* dos vários passos associados a um teste (Figura 6.6).

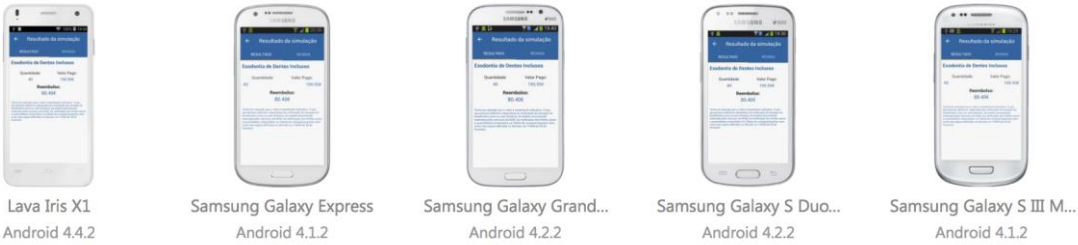

Figura 6.6 - Screenshots de Testes

6.3 Mobile Center

Em qualquer aplicação faz sentido ter *logs* que permitam analisar o comportamento dos utilizadores de modo a perceber que funcionalidades são mais utilizadas tal como analisar porque razão a aplicação não efetuou o comportamento esperado. Para isso, usou-se o *Visual Studio Mobile Center*.

O *Mobile Center* permite a recolha de *logs* de utilização tais como tempo de sessão, número de utilizadores e até causas de *crashes*. Todas as chamadas são geridas a partir do Core, fazendo uso de uma classe estática. Definiu-se um conjunto de eventos específicos (8 no total) para associar a cada *log* (*Getting*, *Got*, *Showing*, *Error*, etc) de modo a que pudesse ser mais fácil identificar grupos de, por exemplo, eventos de erro.

Mesmo sem o envio de eventos, o *Mobile Center* permite obter métricas como por exemplo:

- Localização geográfica dos utilizadores (Figura 6.7).

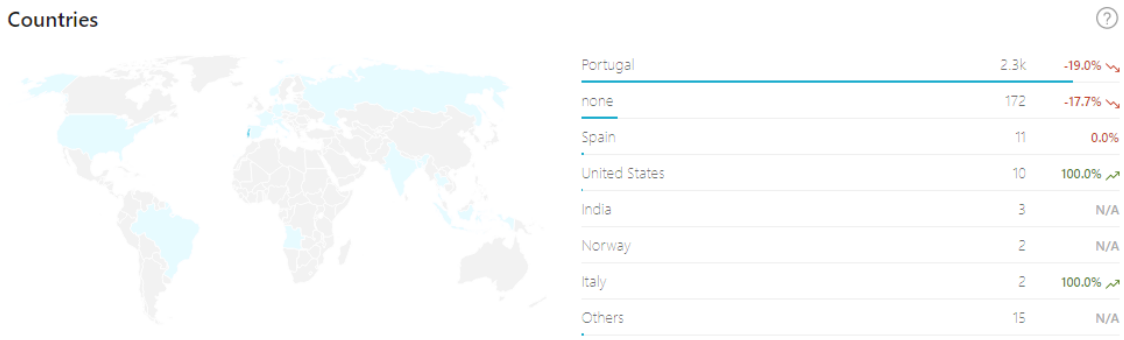

Figura 6.7 - Localização geográfica

- Linguagem dos dispositivos (Figura 6.8).

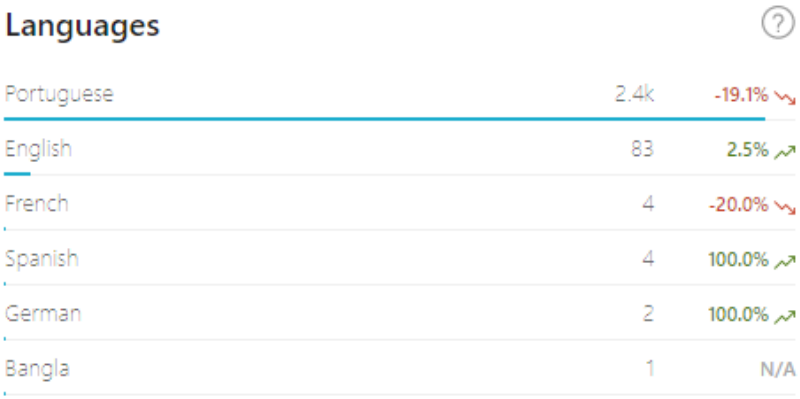

Figura 6.8 - Linguagem dos dispositivos

- Duração das sessões (Figura 6.9).

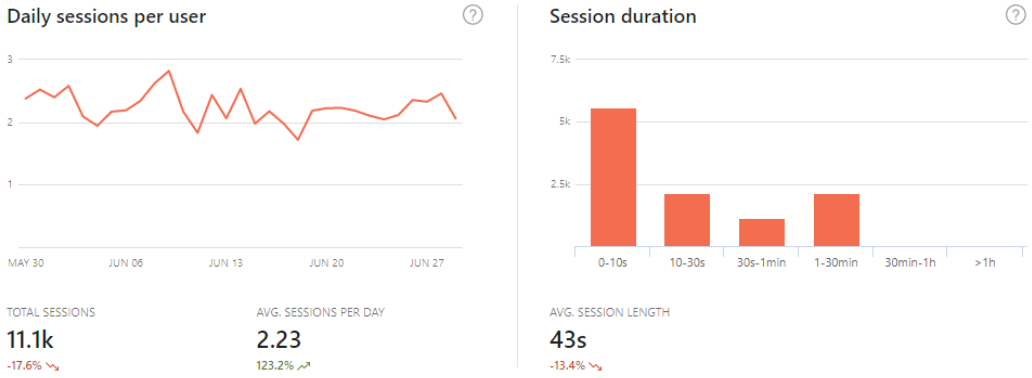

Figura 6.9 - Duração das sessões Mobile Center

- Dispositivos mais usados (Figura 6.10).

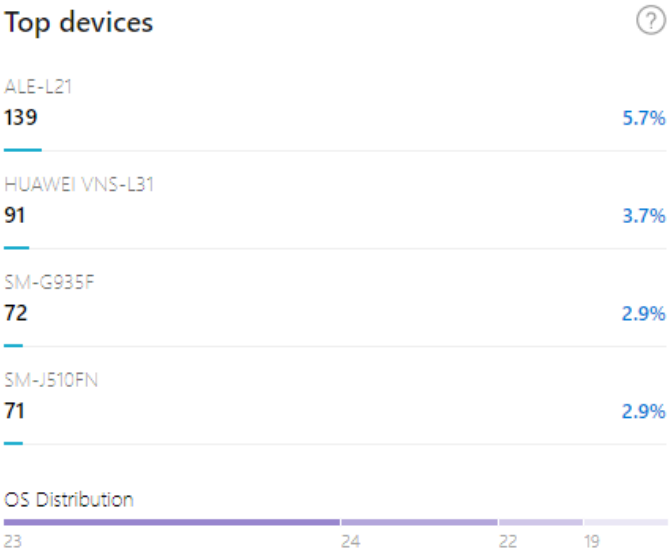

Figura 6.10 - Dispositivos Mobile Center

6.4 Notification Hubs

Como se apostou numa aplicação *cross-platform*, tornou-se necessário arranjar forma de enviar notificações *push*[15] também independente da plataforma. Para tal, usou-se o *Notification Hubs* que trata da gestão dos diferentes serviços de notificações disponíveis a cada plataforma. Para a plataforma de Android foi usado o *Firebase Cloud Messaging* (FCM), enquanto que para iOS o *Apple Push Notification Service* (APNS).

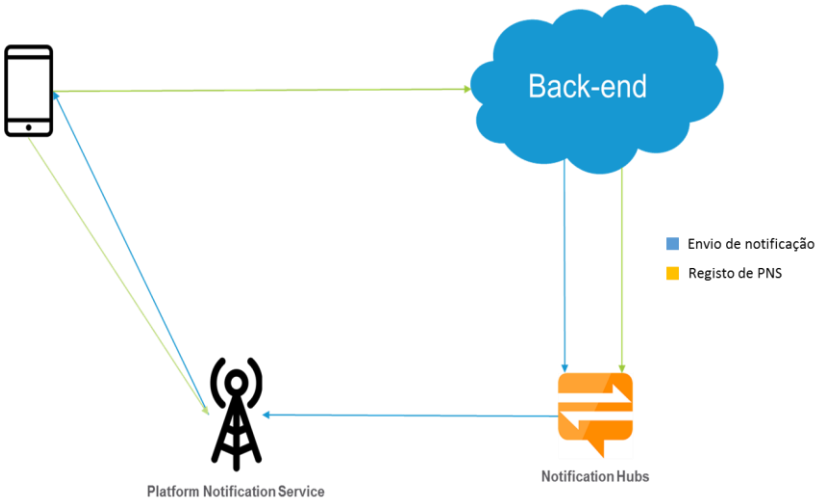

Figura 6.11 - Notification Hubs

Visto que cada plataforma necessita de se registar em diferentes serviços de notificação, foi necessário fazer esse registo de forma nativa, optando-se assim por efetuar esse primeiro registo nas classes de início de cada ambiente. O fluxo após o registo pode observar-se na Figura 6.11: o *Platform Notification Service* (PNS) *Handle* é enviado para o *back-end* de modo a que seja ele a efetuar o registo no *Notification Hubs*. Aquando do envio de uma notificação, o *back-end* efetua um pedido ao *Notification Hubs* que transmite ao PNS correspondente para que seja efetuado o envio da notificação para o(s) dispositivo(s) correspondente(s).

De forma a ser possível enviar notificações específicas a um utilizador foi acordado que, ao efetuar o login, cada utilizador registar-se-ia com um conjunto particular de *tags*. Uma dessas *tags* seria o ID associado ao utilizador. Desta forma quando ocorresse o envio de uma notificação específica saber-se-ia para qual *tag* enviar. De notar que o utilizador ao

[15] Notificações *push* são notificações enviadas diretamente para o dispositivo sem que o utilizador tenha pedido a informação. Funciona como um alerta.

fazer *logout* despoleta um processo que remove a entrada correspondente a esse utilizador na base de dados, tal como elimina o registo da *tag* no *Notification Hubs* para o dispositivo.

6.5 Autenticação

6.5.1 Json Web Tokens

Json Web Tokens (JWT) são *tokens stateless*[16] que permitem de forma rápida e segura a validação da identidade de um utilizador. Faz uso de uma estrutura pré-definida para assegurar que ao decifrar consegue validar que se trata de um *token* válido.

6.5.2 Login

De forma a que a verificação seja feita com um *Authentication Provider* customizado, utilizando os serviços de *back-end* desenvolvidos pelo cliente, o *back-end* em Azure faz *override* ao comportamento *default* e efetua a devida chamada ao serviço disponibilizado no *back-end* do cliente. São enviadas as credenciais de forma segura, esperando pela resposta do lado do cliente para retornar à aplicação.

6.5.3 Segurança

Devido ao uso customizado de JWT, que pela sua natureza são *stateless*, não existe validação em *back-end* em base de dados, sendo que os acessos são de tempo limitado, ou seja, passado um espaço de tempo definido o *token* torna-se inválido.

6.6 Telemetria Azure Search

Telemetria consiste na obtenção de métricas que permite o estudo dos dados adquiridos de modo a auxiliar a tomada de decisão, sendo por isso um conceito importante em qualquer aplicação. Para obtermos telemetria com o Azure *Search* é necessário efetuar uma chamada ao *sdk* do *Application Insights* que guarda os dados que pretendemos recolher. No entanto, o *sdk* do *Application Insights* não está disponível para projetos *.NET Portable*, neste caso PCL. Criou-se então um *controller* do lado do serviço que efetua a chamada com os dados recolhidos do lado da aplicação (Figura 6.12).

[16] *Tokens stateless* são *tokens* que não guardam estado sobre as chamadas passadas.

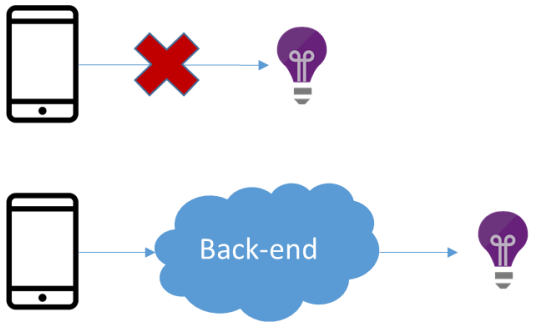

Figura 6.12 - Diagrama de utilização da telemetria do Azure Search

Desta forma tornou-se possível a apresentação ao cliente de um *dashboard* feito em *PowerBI* com algumas métricas importantes como o tipo de pesquisa mais utilizado ou as pesquisas que não geraram *clicks* (Figura 6.13).

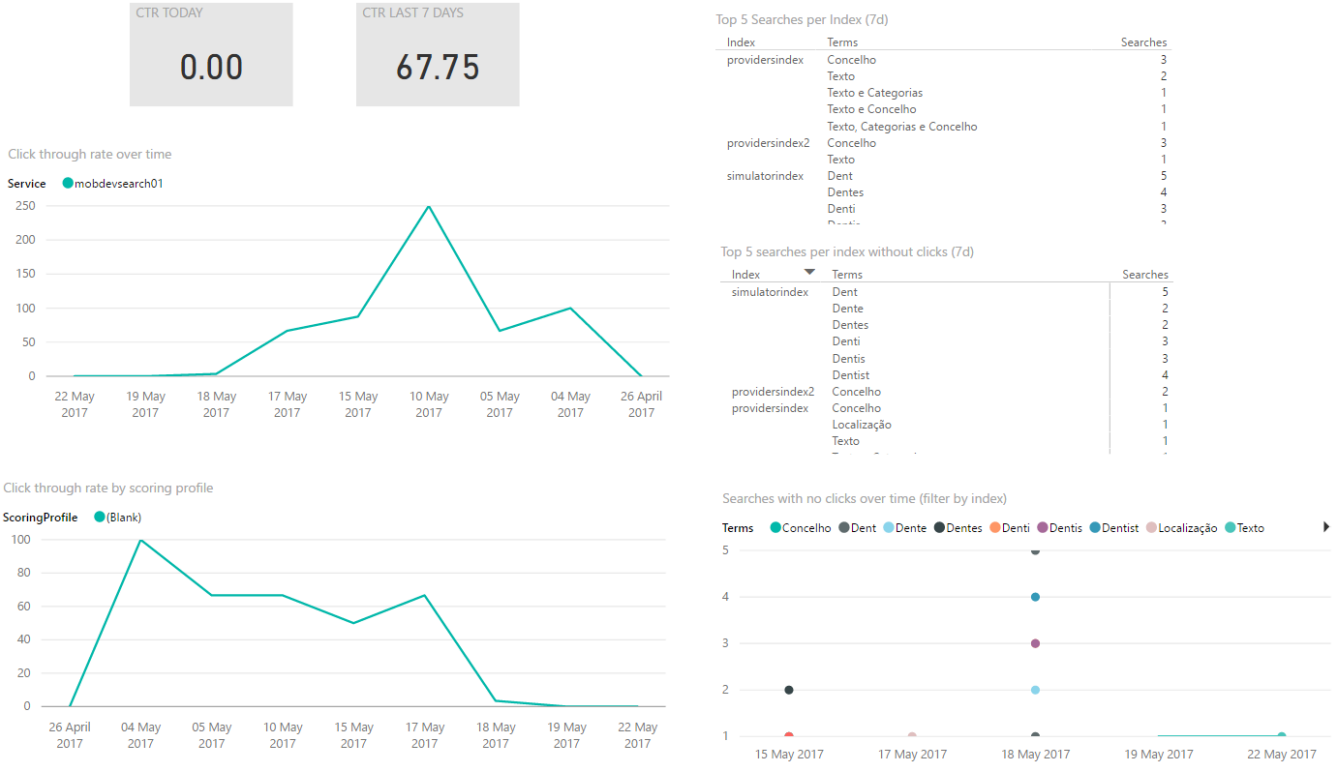

Figura 6.13 - Dashboard AzureSearch

6.7 Controlo de versões

O *back-end* efetua o controlo das diferentes versões da aplicação de modo a ser possível alertar o utilizador de uma possível atualização, atualização essa que pode conter uma correção importante.

Para efetuar o controlo das versões, do lado da aplicação é enviada informação sobre o tipo de sistema operativo e a versão atual da aplicação. No *back-end* existe uma tabela que detalha as combinações *Sistema Operativo - Versão da Aplicação*. Em todas as chamadas

efetuadas ao serviço, a combinação enviada é testada e, caso esteja presente na tabela, é então processada. A atualização dessa tabela é feita de forma manual, paralelamente ao lançamento de uma nova versão da aplicação, independentemente do Sistema Operativo em que é lançada.

Capítulo 7

Conclusões

De modo a conseguir aumentar a base de utilizadores e melhorar a qualidade dos serviços, o cliente procurou expandir o negócio, de modo a também abordar a plataforma *mobile*. Para concretizar esse objetivo fez-se uso de Xamarin e Azure para o desenvolvimento de uma aplicação móvel e do serviço de *back-end* de suporte. Implementou-se usando MVVM e PCL, que permitiu acelerar o desenvolvimento comparativamente com o paradigma de desenvolvimento puramente nativo, com C# como linguagem de programação e o *Visual Studio* como IDE. Graças aos serviços disponíveis na *cloud* foi possível capacitar o canal móvel com um conjunto de funcionalidades *out-of-the-box* disponibilizadas aos utilizadores de forma simples e quase imediata, como uma pesquisa complexa assente em índices de uma base de dados, bem como facilitou a tomada de decisões devido às métricas demonstradas sob a forma de *dashboards* em *PowerBI* ou sob *logs* presentes no *Mobile Center*.

Seguindo a metodologia Agile e fazendo uso de DevOps, garantiu-se uma maior eficiência na gestão de cada *release*, permitindo também que a aplicação entregue correspondesse às espectativas do utilizador, em relação a performance, disponibilidade e disponibilização de novas funcionalidades de forma simples e planeada.

7.1 Problemas Encontrados

Durante a execução do projeto ocorreram alguns problemas:

- Implementação do MVVM / PCL: a implementação do paradigma MVVM provou-se complexa nalguns casos pois algumas das ferramentas usadas para desenvolvimento não se encontram preparadas para lidar com projetos em PCL / MVVM. Tomando o exemplo do *Notification Hubs*, como é necessário efetuar o registo nos respetivos serviços de notificação (FCM e APNS), e cada plataforma disponibiliza componentes externos que expõem os métodos de registo, foi necessário abstrair um pouco do paradigma de modo a cumprir os requisitos de cada plataforma. O mesmo acontece com o registo do dispositivo no *Visual Studio Mobile Center*. Como se atribuiu um ID único de registo para cada plataforma (de modo a

poder ter métricas distintas em ambos os sistemas operativos), provou-se inevitável seguir o processo anterior.

- Refresh de conteúdos após background: foi pedido pelo cliente que a aplicação fizesse a atualização automática dos conteúdos sempre que esta regressasse de *background*. Para tal, implementou-se um algoritmo bastante simples que consiste primeiramente em inicializar o valor de uma variável a zero. Ao parar uma *Activity*, decrementar o valor dessa variável em uma unidade. Ao criar, testar primeiro se essa variável se encontra a zero, caso esteja, significa que a aplicação retornou de background e que é necessário atualizar os conteúdos da mesma. Independentemente da variável se encontrar a zero, o valor da variável é incrementado numa unidade.
- Cache de views do Android: em Android o ciclo de vida dos fragmentos automaticamente faz cache ao necessitar de alocar memória para outras aplicações ou fragmentos. Isto é especialmente útil para dispositivos mais antigos que não possuem tanta RAM. Pode, no entanto, trazer problemas como a "não atualização" dos dados existentes. Conjugando este problema com o fato de ter sido usado PCL, tornou-se impossível diferenciar o tipo de inicialização de cada fragmento na lógica da aplicação. Para tal, e de modo a minimizar o número de chamadas ao serviço, implementou-se um algoritmo semelhante ao de *refresh* dos conteúdos de forma a perceber quando seria necessário efetuar novamente a chamada para atualizar os dados.
- Geração automáticas de builds: a geração de *builds* automáticas começou por ser uma prioridade pois permitiria ao cliente fazer uso das novas funcionalidades sem ser necessário "perder tempo" a gerar executáveis e posteriormente distribuí-los. No entanto o cliente viu alguns entraves e apresentou alguns problemas no uso dessa solução. Optou-se então pela distribuição via *Mobile Center* de cada executável, sendo que o número de *releases* previsto foi consideravelmente diminuído.

7.2 Trabalho futuro

Após a conclusão do projeto, pensaram-se em mais funcionalidades que poderiam fazer sentido implementar e que trariam mais valor à aplicação. A demonstração de estatísticas de utilização da app ao cliente usando *dashboards* de PowerBI, tal como a

utilização de *Cognitive Services*[17] para geração de recomendações, são pontos nos quais nos queremos focar de modo a melhorar os serviços prestados aos utentes.

Quanto à aplicação, são seis as futuras funcionalidades/melhorias:

- Posição Global: um ecrã que representaria custos do utilizador (e dos seus familiares) com o cliente e os descontos recebidos anualmente.
- *Push Notifications* API: serviço que facilite o envio de *push notifications* pelo cliente, de modo a que com a definição de uma interface simples e tratamento centralizado, qualquer sistema de *back-end* do cliente possa enviar notificações.
- Melhoramentos da caixa de mensagens: novas funcionalidades relativas às Mensagens. Englobaria ações como "Marcar como não lida", "Apagar", etc. Estaria também planeado disponibilizar a funcionalidade em offline. Isso traria, no entanto, complicações quanto à segurança pois as mensagens relativas a um utilizador ficariam guardadas no dispositivo, mesmo caso ele efetue *logout.*
- Melhoramentos do Perfil: de momento o perfil consiste na apresentação de uma lista de Chave-Valor em que não são controladas pela App nem as Chaves nem os Valores. A ideia seria encontrar um modelo de dados de modo a que a edição do perfil atualmente implementada pudesse suportar mais regras de negócio, e em função disso adaptar-se o UI.
- Integração com *Wallet*: de momento o cartão virtual é disponibilizado sob forma de uma *view fullscreen*. A ideia seria poder disponibilizar o cartão de forma a que estivesse disponível no *Apple Wallet* em formato pkpass. Desta forma seria possível os titulares efetuarem a partilha com os seus familiares.
- *Slots* de marcação: possibilidade de cada prestador poder disponibilizar *slots* de marcação específicos para utilizadores do cliente, de forma a agilizar o atendimento, tal como a possibilidade de efetuar essa marcação através do canal móvel.

Do lado do *back-end* pensou-se em automatizar o maior número de procedimentos atualmente em curso, tais como:

- *Continuous Integration* (CI): correr as baterias de teste assim que exista um *commit* em *branches* específicos.

[17] *Cognitive Services* é um serviço em Azure que disponibiliza um conjunto de API's ligadas à inteligência artificial de modo a "prever" e oferecer sugestões baseadas nas escolhas dos utilizadores.

- Índices Azure *Search*: reindexação dos índices automaticamente de modo a que não existam discrepâncias entre os dados presentes na base de dados e os dados presentes no *storage* do índice.
- *Continuous Deployment* (CD): geração de executáveis automaticamente de modo a que quando exista um *commit* num *branch* específico possa facilmente ser disponibilizada uma nova versão para o cliente.

7.3 Considerações finais

A experiência do utilizador é o fator chave numa aplicação [35].

Programação em *cross-platform* traz uma plenitude de vantagens como a redução do tempo e do custo de desenvolvimento necessário, a correção de *bugs* ser feita potencialmente apenas uma vez em todas as plataformas disponibilizadas, bem como permite o desenvolvimento partindo de uma linguagem única, podendo fazer uso de *developers* com *skills* de desenvolvimento semelhantes [36, 37, 38].

Existe, no entanto, quem defenda o desenvolvimento nativo pois considera que o utilizador espera de uma aplicação mobile uma utilização excelente, intuitiva e responsiva [39]. É certo que aplicações nativas têm uma performance e uma experiência de utilização maior em relação a aplicações *cross-platform* web [35]. No entanto, quando falamos de aplicações *cross-platform* nativas o resultado já é outro. Aplicações criadas usando, por exemplo, o *Xamarin*, que faz uso das API's nativas de cada plataforma, obtêm um desempenho quase idêntico a uma aplicação nativa, sem fugir a convenções de UI de cada SO, visto ser compilado como uma aplicação nativa. E isso torna-se percetível nas diversas aplicações já criadas por grandes empresas tais como a Sennheiser [40], a FOX Sports, o Crédito Agrícola [41, 42] e até o parlamento do Reino Unido.

O padrão MVVM é usado quando necessitamos de tratar os dados de mais do que uma forma, de modo a que cada *View* possa ser simples [43]. Para além de permitir que tanto o código do *Model*, do *ViewModel* e das *Views* possam rapidamente ser submetidos a testes (unitários no caso do *Model* e do *ViewModel*, UI no caso das *Views*), permite que o código seja mais facilmente modificado (pegando no exemplo de introduzir uma nova implementação para um serviço já existente, apenas mudando uma linha), separando das *Views* toda a lógica do negócio (útil para quando existe uma equipa de programadores a tratar dos *Models* e dos *ViewModels* e outra de designers, focados na criação das *Views*).

Optou-se por usar Xamarin nativo pois nenhum dos programadores tinha experiência em *Xamarin.Forms*. No entanto, através do projeto desenvolvido e da análise efetuada às várias tecnologias e abordagens possíveis para esta implementação, poderíamos ter utilizado *Forms* como uma abordagem mais vantajosa, visto que a aplicação consiste maioritariamente em preenchimento de formulários e exibição de dados, sendo esses os principais pilares de uma aplicação em *Xamarin.Forms*.

Do lado do *back-end* dispúnhamos de duas opções: ou implementaríamos uma *Virtual Machine* (VM) que conseguisse suportar o tráfego gerado, sendo uma opção bastante dispendiosa, ou então usar-se-ia um PaaS. O uso do Azure e o fato de se ter apostado em PaaS ao invés de um serviço físico faz com que o cliente não tenha de gerir uma imensidão de serviços (Figura 7.1), disponibilizando também uma plenitude de serviços *out-of-the-box* como autenticação, bases de dados e funcionalidades *offline*. Para além disso, obtemos uma pletora de vantagens ao utilizar PaaS [19], nomeadamente o suporte de equipas de desenvolvimento geograficamente distantes ou o desenvolvimento de aplicações para variadas plataformas facilmente.

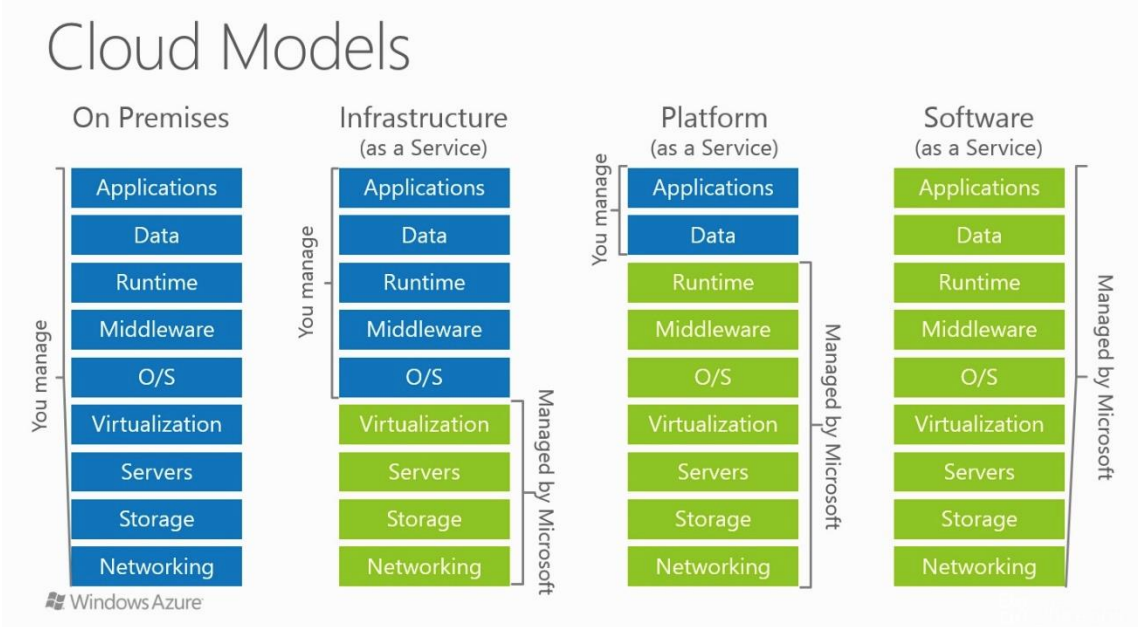

Figura 7.1 - Diferenças entre modelos de Cloud

Quanto mais serviços for necessário o cliente gerir, mais dinheiro irá despender [17]. Desta forma apenas é necessário criar a aplicação em si e fornecer os dados, dados esses presentes nos serviços de *back-end* do cliente.

Depois do término do projeto ficou claro, tanto internamente como pela reação dos utilizadores, que a aplicação ofereceu valor acrescentado ao cliente, isto porque providenciou um meio de comunicação alternativo, disponibilizou novas funcionalidades há

muito pedidas pelos utilizadores e permitiu a aproximação do cliente aos seus utentes. De um ponto de vista de gestão, o projeto permitiu compreender a importância de seguir o pipeline de DevOps. Para garantir uma entrega de qualidade é necessário que cada *release* siga um caminho previamente delineado, pois desta forma os problemas tendem a ser menores devido a entregas contínuas, funcionalidades são planeadas com antecedência, e refinamentos surgem devido à monitorização constante. De notar, no entanto, que algumas tarefas poderiam ter sido delineadas mais cedo, de modo a que os erros fossem reportados gradualmente e o planeamento fosse feito paralelamente ao desenvolvimento e não apenas no início ou entre funcionalidades, isto porque originou atrasos na implementação de novas funcionalidades, e mudanças relativamente a funcionalidades já existentes, fazendo com que a complexidade associada à aprendizagem de DevOps seja recompensada com uma elevada agilidade de negócio transmitida sob a forma de lançamento de mais e melhores versões, com menor esforço por parte de todas as equipas envolvidas.

Finalmente, e mesmo com imprevistos e alterações não planeadas, foi possível cumprir o planeamento e entregar um produto final acima das espectativas do cliente e com uma receção bastante positiva dos utilizadores finais.

7.4 Projetos não relacionados

Durante o decorrer do projeto tive oportunidade de participar noutro projeto durante dois meses que permitiu aprimorar as técnicas de controlo de qualidade. A fase final do projeto consistia numa equipa de 23 pessoas cujo único objetivo residia na realização testes de aceitação para o lançamento de uma nova versão de uma aplicação mobile existente. Neste caso, como apenas nos foi dado acesso à aplicação e não ao código fonte, os testes realizados incidiram apenas sobre UI e lógica para garantir a equiparação de ambos os SO, e sobre a captura de pacotes para garantir a segurança na transmissão de dados.

Garantiu-se também o cumprimento dos requisitos funcionais e todo o suporte aos testes de aceitação do cliente.

Bibliografia

[1] J. Highsmith, “History of Agile,” [Online]. Available: http://agilemanifesto.org/history.html. [Acedido em Maio 2017].

[2] “Agile in a nutshell,” [Online]. Available: http://www.agilenutshell.com/. [Acedido em 10 Junho 2017].

[3] “Different between Scrum and Agile,” [Online]. Available: https://www.provenmethod.com/agile-scrum-whats-difference/. [Acedido em 15 Junho 2017].

[4] M. Marschall, “Scrum vs Agile,” [Online]. Available: http://www.agileweboperations.com/scrum-vs-kanban. [Acedido em 15 Junho 2017].

[5] Agile Alliance, “Agile 101,” [Online]. Available: https://www.agilealliance.org/agile101/. [Acedido em 20 Dezembro 2016].

[6] Microsoft, “Implementing the Model-View-ViewModel pattern,” [Online]. Available: https://msdn.microsoft.com/en-us/library/ff798384.aspx. [Acedido em 20 Junho 2017].

[7] J. Gossman, “MVVM,” [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Acedido em Maio 2017].

[8] J. Gossman, “Introduction to MVVM pattern,” [Online]. Available: https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/. [Acedido em 16 Junho 2017].

[9] “Dependency Injection,” [Online]. Available: https://martinfowler.com/articles/injection.html. [Acedido em Junho 2017].

[10] G. Tiwari, “What is dependency injection?,” 22 Maio 2011. [Online]. Available: https://stackoverflow.com/questions/130794/what-is-dependency-injection/6085922#6085922. [Acedido em 19 Junho 2017].

[11] Microsoft, “The Service Locator Pattern,” [Online]. Available: https://msdn.microsoft.com/en-us/library/ff648968.aspx. [Acedido em 27 Dezembro 2016].

[12] S. Lodge, “Service Location and Inversion of Control,” [Online]. Available: https://github.com/MvvmCross/MvvmCross/wiki/service-location-and-inversion-of-control. [Acedido em Novembro 2016].

[13] M. Fowler, “Martin Fowler,” [Online]. Available: https://martinfowler.com/aboutMe.html. [Acedido em Novembro 2016].

[14] “Inversion of Control,” [Online]. Available: https://martinfowler.com/bliki/InversionOfControl.html. [Acedido em Junho 2017].

[15] M. Fowler, “Inversion of Control Containers and the Dependency Injection pattern,” 23 Janeiro 2004. [Online]. Available: http://www.martinfowler.com/articles/injection.html. [Acedido em Novembro 2016].

[16] J. Abrahamsson, “Inversion of Control – An Introduction with Examples in .NET,” Março 2013. [Online]. Available: http://joelabrahamsson.com/inversion-of-control-an-introduction-with-examples-in-net/. [Acedido em Novembro 2016].

[17] “Azure: On Premises vs IaaS vs PaaS vs SaaS,” 21 Maio 2015. [Online]. Available: https://stack247.wordpress.com/2015/05/21/azure-on-premises-vs-iaas-vs-paas-vs-saas/. [Acedido em 29 Dezembro 2016].

[18] Apprenda, “IaaS, PaaS, SaaS (Explained and Compared),” [Online]. Available: https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/. [Acedido em 27 Dezembro 2016].

[19] Microsoft, “O que é PaaS?,” [Online]. Available: https://azure.microsoft.com/pt-pt/overview/what-is-paas/. [Acedido em 20 Dezembro 2016].

[20] “Xamarin,” [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/f/f2/Xamarin-logo.svg/1200px-Xamarin-logo.svg.png. [Acedido em 25 Junho 2017].

[21] Microsoft, “Microsoft Azure,” [Online]. Available: https://azure.microsoft.com/svghandler/dns/?width=600&height=315. [Acedido em 25 Junho 2017].

[22] “Visual Studio,” [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e4/Visual_Studio_2013_Logo.svg/990px-Visual_Studio_2013_Logo.svg.png. [Acedido em 25 Junho 2017].

[23] CodeMentor, “C#,” [Online]. Available: https://cdn.codementor.io/assets/tutors/c-sharp-tutors-online.png. [Acedido em 25 Junho 2017].

[24] Microsoft, “Visual Studio Mobile Center,” [Online]. Available: https://www.visualstudio.com/pt-br/vs/visual-studio-mobile-center/. [Acedido em 28 Dezembro 2016].

[25] IDC, “Smartphone OS Market Share, 2016 Q3,” [Online]. Available: http://www.idc.com/promo/smartphone-market-share/os. [Acedido em Dezembro 2016].

[26] “What is C#,” [Online]. Available: https://www.computerhope.com/jargon/c/csharp.htm. [Acedido em Junho 2017].

[27] Microsoft, “Introduction to C#,” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework. [Acedido em Maio 2017].

[28] Microsoft, “What is Microsoft Azure?,” [Online]. Available: https://azure.microsoft.com/pt-pt/overview/what-is-azure/. [Acedido em 28 Dezembro 2016].

[29] DotNetTricks, “Xamarin native vs Xamarin.Forms,” [Online]. Available: http://www.dotnettricks.com/img/xamarin/xamarin-vs-xamarin-forms.png. [Acedido em 15 Junho 2017].

[30] Facebook, “React Native,” [Online]. Available: https://facebook.github.io/react-native/. [Acedido em 25 Junho 2017].

[31] OutSystems, “OutSystems,” [Online]. Available: https://www.outsystems.com/. [Acedido em 25 Junho 2017].

[32] C. O'Sullivan, “Navigation Patterns,” 15 Abril 2015. [Online]. Available: https://cms-assets.tutsplus.com/uploads/users/810/posts/23616/image/navigation.png. [Acedido em 10 Junho 2017].

[33] Microsoft, “What are NSGs,” [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-nsg. [Acedido em Junho 2017].

[34] TeamTeam.Net, “Xamarin Test Recorder,” 6 Dezembro 2015. [Online]. Available: https://i2.wp.com/teamtam.net/wp-content/uploads/2015/12/XamarinTestRecorder.png. [Acedido em 25 Junho 2017].

[35] Y Media Labs, “Hybrid vs Native Mobile Apps – The Answer is Clear,” 9 Setembro 2016. [Online]. Available: https://www.ymedialabs.com/hybrid-vs-native-mobile-apps-the-answer-is-clear/. [Acedido em 17 Dezembro 2016].

[36] “Pros and Cons of Cross-Platform Mobile App Development,” [Online]. Available: https://www.infoq.com/articles/mobile-cross-platform-app-development. [Acedido em Novembro 2016].

[37] “Native vs Hybrid vs Web Application,” [Online]. Available: http://theninehertz.com/native-hybrid-web-applications. [Acedido em Novembro 2016].

[38] Telerik, “Hybrid vs Native,” [Online]. Available: http://docs.telerik.com/platform/help/getting-started/hybrid-or-native. [Acedido em Dezembro 2016].

[39] “Native vs Cross-platform,” [Online]. Available: https://yalantis.com/blog/native-vs-cross-platform-app-development-shouldnt-work-cross-platform/. [Acedido em Novembro 2016].

[40] Xamarin, “Media and Xamarin,” [Online]. Available: https://www.xamarin.com/customers/media. [Acedido em 26 Dezembro 2016].

[41] Xamarin, “Crédito Agrícola,” [Online]. Available: https://s3.amazonaws.com/www.xamarin.com-assets/case-studies/Credito+Agricola.pdf. [Acedido em 26 Dezembro 2016].

[42] Xamarin, “Finance and Xamarin,” [Online]. Available: https://www.xamarin.com/customers/finance. [Acedido em 26 Dezembro 2016].

[43] “Why use Mvvm?,” [Online]. Available: http://stackoverflow.com/questions/2653096/why-use-mvvm. [Acedido em 27 Dezembro 2016].

[44] “Advanced IDE for iOS and Android,” [Online]. Available: https://www.xamarin.com/studio. [Acedido em Novembro 2016].

[45] M. Tulloch, Introducing Windows Azure, Redmond, Washington: Microsoft Press, 2013.

[46] “Which is better, native app development vs. cross platform app?,” [Online]. Available: https://www.quora.com/Which-is-better-native-app-development-vs-cross-platform-app. [Acedido em Novembro 2016].

[47] “Cross-platform Mobile Development VS Native Development,” [Online]. Available: https://program-ace.com/press-room/articles/cross-platform-vs-native-development. [Acedido em Novembro 2016].

[48] “Lazy Loading,” [Online]. Available: https://www.codeproject.com/Articles/652556/Can-you-explain-Lazy-Loading. [Acedido em Novembro 2016].

[49] Dynatrace, “Mobile Apps: What Consumers Really Need and Want,” [Online]. Available: https://info.dynatrace.com/rs/compuware/images/Mobile_App_Survey_Report.pdf. [Acedido em 26 Dezembro 2016].

[50] Jeff, “Xamarin vs. Native App Development,” 8 Fevereiro 2016. [Online]. Available: http://willowtreeapps.com/blog/xamarin-vs-native-app-development/. [Acedido em 26 Dezembro 2016].

[51] “What is a singleton in C#?,” 28 Janeiro 2010. [Online]. Available: http://stackoverflow.com/questions/2155688/what-is-a-singleton-in-c. [Acedido em 20 Dezembro 2016].

[52] C. Warren, “The Pros and Cons of Cross-Platform App Design,” 16 Fevereiro 2016. [Online]. Available: http://mashable.com/2012/02/16/cross-platform-app-design-pros-cons/#XdzTxM_zQGq9. [Acedido em 17 Dezembro 2016].

[53] C. Williams, “Xamarin vs Native,” 27 Setembro 2016. [Online]. Available: http://www.colbylwilliams.com/2016/09/27/xamarin-vs-native.html. [Acedido em 17 Dezembro 2016].

[54] “Why do I need an IoC container as opposed to straightforward DI code?,” [Online]. Available: http://stackoverflow.com/questions/871405/why-do-i-need-an-ioc-container-as-opposed-to-straightforward-di-code. [Acedido em 27 Dezembro 2016].

[55] Microsoft, “Design of the SharePoint Service Locator,” [Online]. Available: https://msdn.microsoft.com/en-us/library/ff648949.aspx. [Acedido em 27 Dezembro 2016].

[56] “Why should i use Mvvm in Silverlight app?,” [Online]. Available: http://stackoverflow.com/questions/4622195/why-should-i-use-mvvm-in-silverlight-app/4622709#4622709. [Acedido em 27 Dezembro 2016].

[57] “When not to use Mvvm?,” [Online]. Available: http://stackoverflow.com/questions/4648057/when-not-to-use-mvvm. [Acedido em 27 Dezembro 2016].

[58] A. Rose, “UX designers: Side drawer navigation could be costing you half your user engagement,” 8 Abril 2014. [Online]. Available: http://thenextweb.com/dd/2014/04/08/ux-designers-side-drawer-navigation-costing-half-user-engagement/. [Acedido em 27 Dezembro 2016].

[59] Google, “Dev and Beta Update for Chrome OS,” 2 Novembro 2012. [Online]. Available: https://chromereleases.googleblog.com/2012/11/dev-and-beta-update-for-chrome-os.html. [Acedido em 27 Dezembro 2016].

[60] B. Kepes, “Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS,” [Online]. Available: https://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/. [Acedido em 27 Dezembro 2016].

[61] J. Archer, “The hamburger menu doesn’t work,” [Online]. Available: http://jamesarcher.me/hamburger-menu. [Acedido em 27 Dezembro 2016].

[62] A. Favell, “Should you use a hamburger icon on your mobile menu?,” 27 Julho 2016. [Online]. Available: https://searchenginewatch.com/2016/07/27/should-the-hamburger-icon-be-on-your-mobile-menu/. [Acedido em 10 Dezembro 2016].

[63] K. Pernice e R. Budiu, “Hamburger Menus and Hidden Navigation Hurt UX Metrics,” 26 Junho 2016. [Online]. Available: https://www.nngroup.com/articles/hamburger-menus/. [Acedido em 17 Dezembro 2016].

[64] “Apple on Hamburger Menus,” 30 Junho 2014. [Online]. Available: http://blog.manbolo.com/2014/06/30/apple-on-hamburger-menus. [Acedido em 27 Dezembro 2016].

[65] R. Kumar, “Why We Banished the Hamburger Menu From Our iPhone App,” 13 Janeiro 2015. [Online]. Available: https://redbooth.com/blog/hamburger-menu-iphone-app. [Acedido em Novembro 2016].

[66] L. Wroblewski, “Obvious Always Wins,” 27 Abril 2015. [Online]. Available: http://www.lukew.com/ff/entry.asp?1945. [Acedido em Novembro 2016].

[67] S. Hoober, “Why It’s Totally Okay to Use a Hamburger Icon,” 4 Maio 2015. [Online]. Available: http://www.uxmatters.com/mt/archives/2015/05/why-its-totally-okay-to-use-a-hamburger-icon.php. [Acedido em Dezembro 2016].

[68] “The Agile Movement,” 23 Outubro 2008. [Online]. Available: http://agilemethodology.org/. [Acedido em 28 Dezembro 2016].

[69] Microsoft, “Azure Search,” [Online]. Available: https://azure.microsoft.com/pt-pt/services/search/. [Acedido em 21 Dezembro 2016].

[70] M. Fowler, “Service Locator,” [Online]. Available: https://martinfowler.com/articles/injection.html#UsingAServiceLocator. [Acedido em Maio 2017].

[71] Auth0, “What are JWT?,” [Online]. Available: https://jwt.io/. [Acedido em 15 Junho 2017].

[72] C. O'Sullivan, “Designing for both Android and iOS,” 15 Abril 2015. [Online]. Available: https://webdesign.tutsplus.com/articles/a-tale-of-two-platforms-designing-for-both-android-and-ios--cms-23616. [Acedido em 10 Junho 2017].

[73] D. Brown, “What is DevOps?,” [Online]. Available: http://donovanbrown.com/post/what-is-devops. [Acedido em 15 Junho 2017].

[74] M. Httermann, DevOps for Developers, 2012.