



UNIVERSIDADE
LUSÓFONA

Plataforma de Gestão de Pedidos

Trabalho Final de Curso

Nome do Aluno: Daniel Silva

a21207972

Nome do Orientador: Bruno Cipriano

Trabalho Final de Curso | LEI | 27 de Fevereiro de 2016

www.ulusofona.pt

Direitos de cópia

Plataforma de Gestão de Pedidos, Copyright de Daniel Silva, ULHT.

A Escola de Comunicação, Arquitectura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Índice Geral

Índice de Figuras	5
Resumo.....	6
Abstract	7
1 Introdução	8
2 Enquadramento teórico	9
2.1 Tecnologias	9
2.1.1 WebAPI.....	9
2.1.2 Mono	9
2.1.3 ASP.NET Identity	9
2.1.4 Xamarin.....	9
2.1.5 ASP.Net MVC.....	9
2.1.6 Azure	10
2.1.7 Visual Studio Team Services	10
2.1.8 SQL Server.....	10
3 Método	11
3.1 Análise.....	11
3.2 Arquitetura do Sistema.....	11
3.2.1 Modelo de Dados	12
3.2.2 WebAPI.....	17
3.3 Infraestrutura necessária.....	18
4 Resultados	19
5 Conclusões e trabalho futuro.....	24
6 Bibliografia	25
7 Glossário	25
ANEXO A Requisitos	I
Requisitos funcionais	II
Requisitos não funcionais.....	III
ANEXO B Integração com Google Maps (guia para programadores).....	IV

Índice de Figuras

Figura 1 - Arquitetura do sistema	12
Figura 2 - Relação entre entidades - Produto	12
Figura 3 - Relação entre entidades e pedidos.....	13
Figura 4 - Relação entre entidades e utilizador	14
Figura 5 - Relação entre utilizador e localização.....	15
Figura 6 - Diagrama de Modelo de Dados.....	16
Figura 7 - Infraestrutura para um único equipamento	18
Figura 8 - Infraestrutura para um Mac OS centralizado.....	19
Figura 9 - Configuração do projeto Android - Separador “General”	20
Figura 10 - Configuração do projeto Android - Separador “Advanced”	21
Figura 11 - Vista disponibilizada a um utilizador/operador de loja	21
Figura 12 - Ecrã de visualização dos locais de uma empresa	22
Figura 13 - Ecrã inicial da Aplicação móvel em iOS	22
Figura 14 - Ecrã inicial da Aplicação móvel em Android	23
Figura 15 - Resultado integração API Google Maps	VI

Resumo

Este trabalho final de curso tem como objetivo a criação de uma plataforma, que permita a criação de pedidos, via *WebAPI*, sobre determinados produtos e disponibilizar um *backoffice* para gestão dos mesmos.

A ideia de os pedidos serem criados via *WebAPI*, tem como objetivo potenciar o uso da plataforma com sistemas já existentes, seja na vertente aplicação móvel ou aplicação *web*.

De forma a conseguir passar a ideia de forma mais clara, foi contemplada a demonstração de uma implementação para equipamentos móveis com o objetivo de explorar a plataforma de desenvolvimento *Xamarin*. Esta é uma aplicação móvel que tem também como objetivo conseguir visualizar a lista de produtos configurada (de um cliente fictício), efetuar pedidos de um ou vários produtos, e ter um *feedback* visual e textual do estado do mesmo.

O *backoffice* permite a um operador alterar o estado de um determinado pedido, de forma a que o cliente consiga perceber em que estado está, ver estatísticas dos pedidos realizados e/ou pendentes, criar utilizadores, empresas, grupos de empresas e produtos (com ou sem categorias).

Palavras-Chave: Gestão de Pedidos, Aplicação Móvel, *Xamarin*, *WebAPI*, *Backoffice*

Abstract

The objective of this final Project, is to develop an independent platform to manage orders from clients and provide an interface via WebAPI to create orders from configured products.

The idea of using WebAPI is to allow current mobile apps and other web application to integrate with this platform, with some simplicity.

To be able to show the full potential of this platform, a mobile application was developed to show what can we do with the platform, by allow someone to create an order, based on one or more products, and get feedback from the status of the order.

The backoffice allows an employee to change the status of the order, view statistics of all orders, create users, companies, company groups and products (with or without categories).

Keywords: Order Management, Mobile, Xamarin, WebAPI, Backoffice

1 Introdução

Este projeto foi desenvolvido no âmbito do Trabalho Final de Curso, na Licenciatura em Engenharia Informática da universidade Lusófona de Humanidades e Tecnologias.

Nos dias de hoje, é imprescindível a qualquer empresa a presença online, seja da forma *Web* e/ou aplicações móveis. Neste sentido, a proposta deste projeto assenta num problema que já detetei em alguns estabelecimentos comerciais (setor da restauração), que reside na falta de uma componente móvel e web site, para que os clientes possam efetuar encomendas à distância, evitando assim esperas desnecessárias nos estabelecimentos.

Com este sistema, consegue-se melhorar a relação com os clientes, já que estes têm a possibilidade de aproveitar melhor o tempo.

Embora o projeto em si não tenha a componente móvel como objetivo principal, achei importante colocar uma demonstração do mesmo, de forma a mostrar um pouco as possibilidades que temos de interação com os clientes, usando *push notifications*. De forma a conseguir ter uma aplicação multiplataforma, esta será desenvolvida usando *Xamarin*, recentemente integrado no mundo Microsoft e já com muitas provas dadas no mercado.

Toda esta integração de plataformas e diferente *hardware* é algo que para mim é motivador, dado que obriga a explorar um mundo que ainda não tenho o contacto muito sólido, mas que espero num futuro próximo ser uma realidade diária.

2 Enquadramento teórico

Este capítulo apresenta as tecnologias utilizadas no decorrer do projeto, e de maior importância.

2.1 Tecnologias

2.1.1 WebAPI

Uma WebAPI é uma interface de comunicação utilizada entre sistemas, normalmente os formatos podem variar entre JSON e XML, e surgiu com a Web 2.0. Nos últimos anos tem havido uma crescente utilização das WebAPI's, principalmente pelos grandes *players* das tecnologias, e isso ajudou a potenciar esta forma de integração de aplicações. Atualmente as WebAPI têm uma utilização que pode ir desde um *website*, que com Javascript faz acesso a uma WebAPI, a uma aplicação móvel nativa que através de rotinas próprias consegue também ter acesso a WebAPI's.

2.1.2 Mono

O Mono¹ é uma plataforma de desenvolvimento baseada na *framework* .Net da Microsoft, que permite criação de aplicações multiplataforma recorrendo à linguagem C Sharp (C#).

2.1.3 ASP.NET Identity

É um sistema que disponibiliza mecanismos de autenticação, gestão de utilizadores e *roles*. Este sistema está integrado na *framework* .Net, permite integração com outros provedores de autenticação (Facebook, Google e outros através de customização).

2.1.4 Xamarin

O Xamarin é uma plataforma de desenvolvimento baseada em Mono, que atualmente, permite desenvolver software para equipamentos móveis, usando a *suite* própria (Xamarin Studio) ou integrado no Visual Studio da Microsoft. É uma plataforma que tem tido um crescimento a nível de utilização, e recentemente foi adquirida pela Microsoft, passando a estar disponível de forma gratuita na versão do Visual Studio Community. Desta forma, esta plataforma motivou a sua utilização dada a possibilidade da multiplataforma.

2.1.5 ASP.Net MVC

O ASP.Net é mais uma plataforma da Microsoft, que já conta com mais de uma década de existência, mais recentemente foi acrescentada uma nova arquitetura de

¹ <http://www.mono-project.com>

desenvolvimento, o MVC, bastante utilizada no desenvolvimento de aplicações Web, dada a organização que impõe ao desenvolvimento.

2.1.6 Azure

A Azure² é uma plataforma *cloud* da Microsoft, que disponibiliza um vasto leque de ferramentas e serviços, que vão desde o alojamento de sites, bases de dados, computação de alto desempenho, interligação de infraestrutura *cloud* com infraestrutura *on-premise*, e com a vantagem de não ter custo de aquisição de equipamentos, pagando apenas pelo tempo de uso.

2.1.7 Visual Studio Team Services

O Visual Studio Team Services³ é um serviço relativamente recente, que permite aos programadores (individuais ou corporativos), a integração com o Visual Studio, permitindo assim ter um repositório de código (com controlo de versões), integração continua (*continuous integration*) e execução de testes automáticos. Além destas funcionalidades base, têm a possibilidade de integrar com a Azure, de forma a por exemplo permitir a publicação de versões de forma automatizada. No âmbito do projeto, este serviço foi apenas utilizado como repositório de código. Um pormenor interessante em relação ao Visual Studio Team Services é a possibilidade de se utilizar `git` como gestor de versionamento de código fonte, tornando assim este serviço muito versátil.

2.1.8 SQL Server

Este é um dos sistemas de gestão de bases de dados mais conhecidos no mundo de IT, estando presente na maioria das infraestruturas das empresas, onde a tecnologia Microsoft é utilizada. Além de bases de dados, o pacote SQL Server contempla também outras ferramentas de integração, *reporting* e análise de dados.

² <https://portal.azure.com>

³ <https://www.visualstudio.com/team-services/>

3 Método

3.1 Análise

Apesar de a ideia ter sugerida por mim, foi necessário pensar nos requisitos que eram indispensáveis, deixando de lado alguns requisitos mais elaborados que, embora tornassem o sistema mais interessante, implicariam certamente nos prazos de entrega.

Para facilitar o desenvolvimento foram então enumerados os requisitos funcionais e não funcionais, categorizados por plataforma (*Backoffice* e *Mobile*) e no caso dos requisitos funcionais, o perfil que corresponde cada um dos requisitos.

Os requisitos definidos e acordados com o Professor orientador (Bruno Cipriano) estão enumerados no Anexo A.

3.2 Arquitetura do Sistema

O sistema desenvolvido, é constituído por três componentes de alto nível: *Backoffice*, *WebAPI* e aplicações móveis. O funcionamento do sistema é totalmente online, não estando prevista a opção offline, e está totalmente integrado na Azure, embora qualquer outra plataforma *Cloud* possa alojar o sistema (desde que seja compatível com .Net MVC e SQL Server). Esta escolha deveu-se ao facto de ser possível o uso gratuito através das contas DreamSpark⁴, que fazem parte de um acordo entre a Universidade Lusófona e a Microsoft.

No âmbito do sistema *Cloud*, este fornece-nos o Servidor Web, Base de Dados e o certificado SSL para uso do HTTPS, que é um requisito para o acesso de dados através de aplicações iOS (Apple).

A comunicação entre as aplicações móveis e a *WebAPI* é assente em pedidos HTTP REST, com formato JSON. Estes pedidos são interpretados por cada uma das componentes, tendo sido apenas necessário implementar a conversão dos dados na componente *Mobile*, usando sempre funções nativas da *framework* .Net.

⁴ <http://dreamspark.com/>

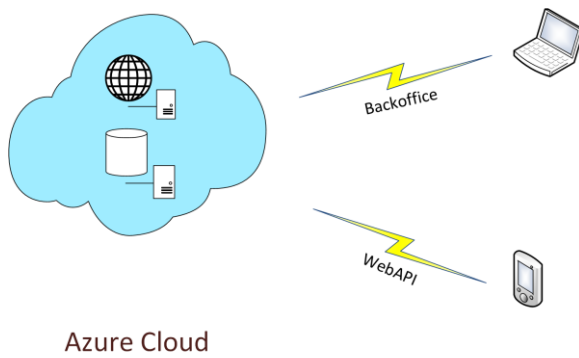


Figura 1 - Arquitetura do sistema

3.2.1 Modelo de Dados

De forma a ir de encontro aos requisitos estabelecidos, foi criado um modelo de dados que represente as entidades e a forma que estas se relacionam.

Em termos de regras, foi definido o seguinte:

- Um produto tem de ter uma categoria e pertencer a uma empresa.

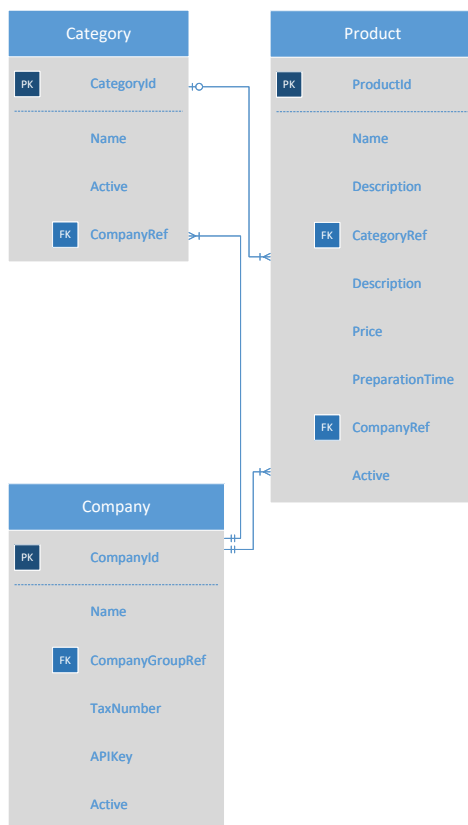


Figura 2 - Relação entre entidades - Produto

- Um pedido é constituído por um ou vários produtos, uma localização e está relacionado com uma empresa

Apesar de os produtos terem já esta relação com a empresa, não é viável em termos de carga e complexidade a pesquisa de pedidos de uma determinada empresa, daí que a solução foi também passar esta relação no pedido.

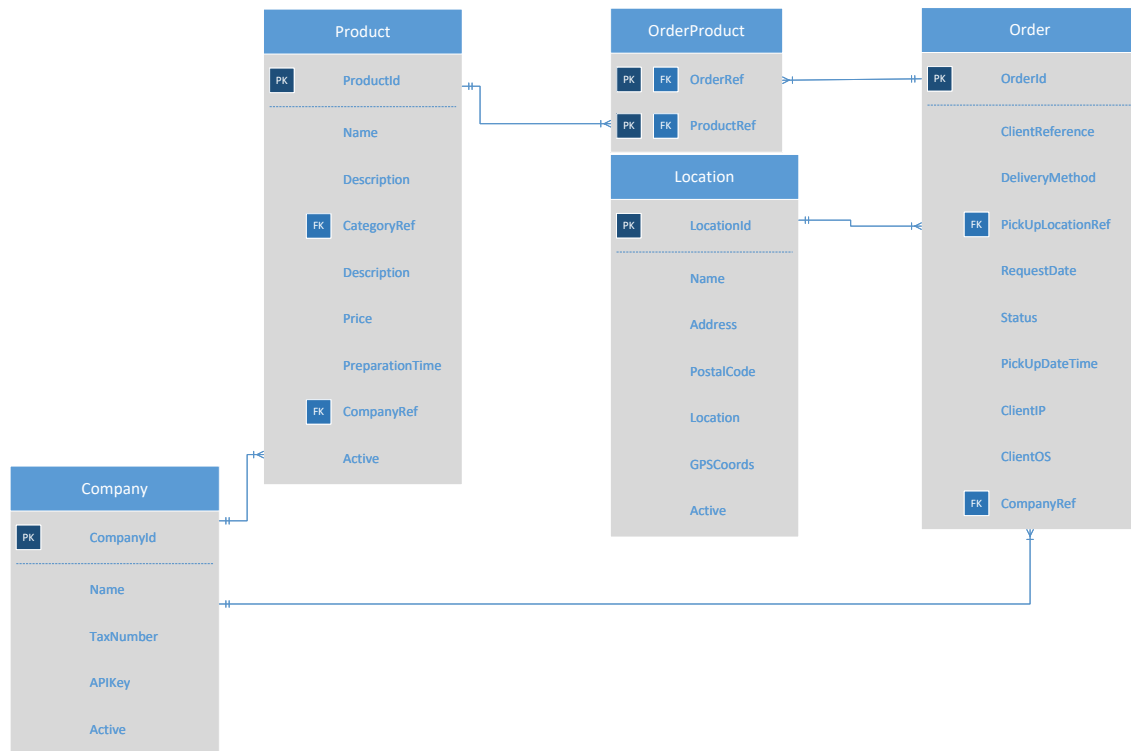


Figura 3 - Relação entre entidades e pedidos

- Um utilizador está associado a uma ou mais empresas

Este cenário implicou uma relação entre dois modelos de dados, dado que foi utilizado o ASP.Net Identity da Microsoft para a gestão de utilizadores, e esta já possui um modelo próprio. A dependência ficou assente no ID do utilizador.

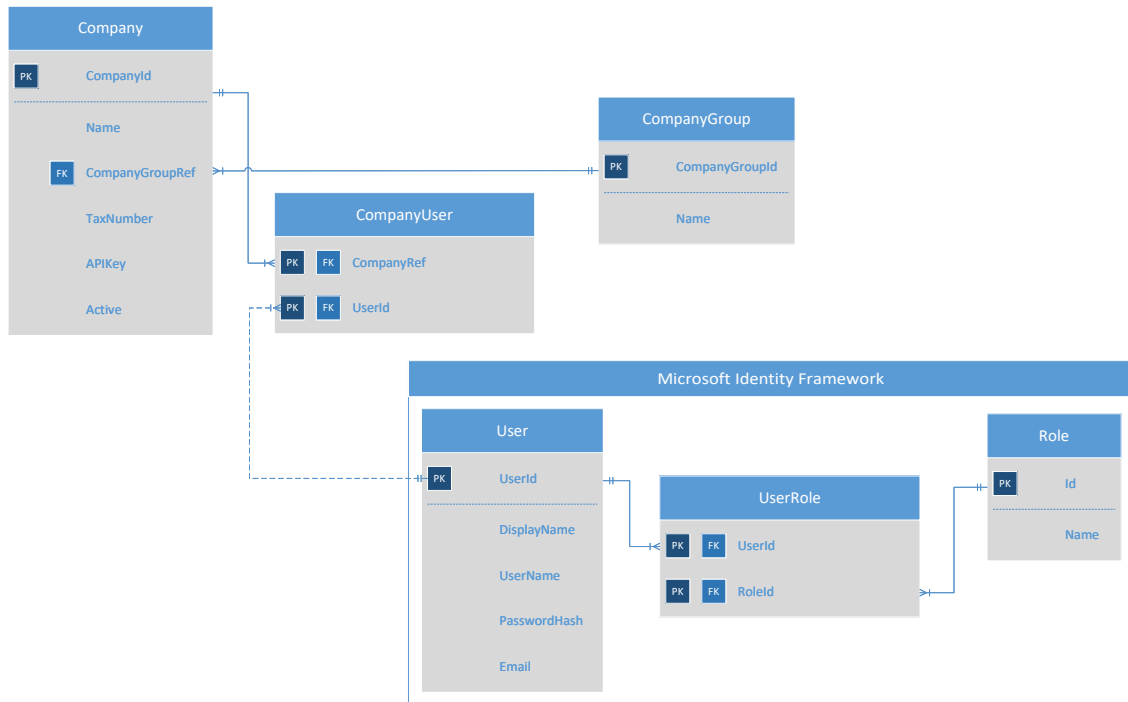


Figura 4 - Relação entre entidades e utilizador

- Um utilizador tem um localização

Cada utilizador está assignado a uma localização, por forma a ver os pedidos sobre a mesma, portanto foi necessário criar uma tabela de suporte a esta relação. Apesar de ser possível ter o mesmo utilizador em várias localizações, a restrição de um utilizador, uma localização, é garantida pela própria aplicação:

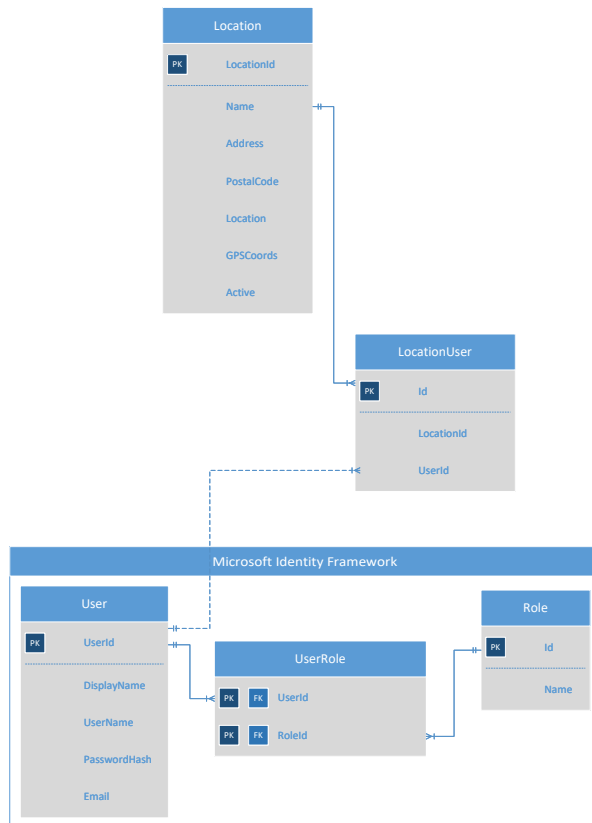


Figura 5 - Relação entre utilizador e localização

A imagem da figura 5 representa o modelo de dados na sua totalidade, estando incluído o modelo de dados externo representativo do ASP .Net Identity.



Figura 6 - Diagrama de Modelo de Dados

3.2.2 WebAPI

A WebAPI disponibilizada pela plataforma permite que as aplicações móveis comuniquem com o sistema, permitindo desta forma diversas operações. Em seguida, são enumeradas as operações disponibilizadas, a sua função e que parâmetros são necessários e opcionais.

GetProducts		
Obtenção da lista de produtos para uma empresa, opcionalmente filtra a categoria dos produtos.		
Parâmetro	Descrição	Obrigatório
companyId	Identificação da empresa	Sim
categoryId	Identificação da categoria do produto	Não

GetCategories		
Obtenção da lista de categorias para uma empresa.		
Parâmetro	Descrição	Obrigatório
companyId	Identificação da empresa	Sim

GetRequests		
Obtenção da lista de pedidos de um utilizador.		
Parâmetro	Descrição	Obrigatório
clientReference	Identificação do cliente	Sim

GetCategories		
Obtenção da lista de categorias de uma empresa.		
Parâmetro	Descrição	Obrigatório
companyId	Identificação da empresa	Sim

GetLocations		
Obtenção da lista de localizações de uma empresa.		
Parâmetro	Descrição	Obrigatório
companyId	Identificação da empresa	Sim

CreateRequest		
Criação de um novo pedido.		
Parâmetro	Descrição	Obrigatório
companyId	Identificação da empresa	Sim
clientReference	Identificação do cliente	Sim
locationId	Identificação do local para levantamento do pedido	Sim
pickupTicks	Data de levantamento, no format Ticks (1 tick representa 100 nanosegundos)	Sim
products	Lista de produtos escolhidos	Sim

3.3 Infraestrutura necessária

Durante o desenvolvimento deste sistema, foi possível ter uma melhor ideia do que é necessário (em termos de hardware e software) para o desenvolvimento de um *Backoffice* e aplicação móvel multiplataforma.

Numa primeira abordagem, comecei por utilizar o Visual Studio, com Windows 10, onde foi desenvolvido todo o *Backoffice*, a classe partilhada de acesso a dados via WebAPI (para mobile) e a aplicação para Android. A ideia inicialmente seria utilizar somente um computador com Windows e Visual Studio para o desenvolvimento, mas um dos requisitos para desenvolver para iOS é ter um equipamento com Mac OS. Este equipamento tem duas funções principais: Compilação do código e execução do simulador para teste da Aplicação a desenvolver.

Após iniciar o desenvolvimento da *App* para o iOS, rapidamente me apercebi que poderia ser melhor ter acesso ao simulador na mesma máquina que desenvolvia, mas para tal teria de abdicar do Visual Studio, já que este não existe para Mac OS (tirando a versão mais simples Visual Studio Code, mas que tem outro fim). Em alternativa, a plataforma *Xamarin* disponibiliza uma ferramenta de edição de código bastante completa, chamada de *Xamarin Studio*. Com esta ferramenta é possível integrar com Visual Studio Team Services (usando *git*), e abrir os mesmos projetos que temos num computador Windows, estando limitado a projetos do tipo: *Class Library*, *App iOS* e *App Android*.

Em termos de cenários em concreto, deixo duas das melhores possibilidades para o desenvolvimento com *Xamarin* para a três plataformas:

Um equipamento físico com Mac OS (requere 16GB de memória RAM para o sistema ser fluído)

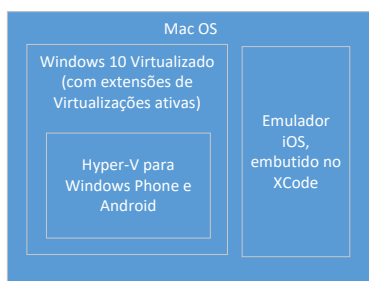


Figura 7 - Infraestrutura para um único equipamento

Dois equipamentos físicos, um com Mac OS e outro com Windows 10.

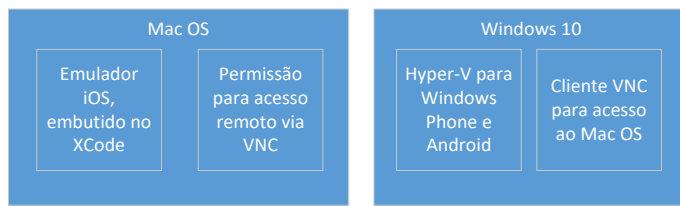


Figura 8 - Infraestrutura para um Mac OS centralizado

Para um único programador, o primeiro cenário é o ideal, já que em termos de custo é inferior ao segundo e seguramente mais cómodo. Já o segundo cenário está mais adaptado a um ambiente em que existem várias pessoas a trabalhar no mesmo projeto *Xamarin* (o Mac OS fica como único equipamento na rede e passam a existir vários computadores com Windows 10), pelo que o custo será inferior em termos de infraestrutura e continua a possibilitar a multiplataforma.

4 Resultados

Com este trabalho foi possível aferir o potencial do desenvolvimento de *App's* com *Xamarin* e a forma de explorar a integração de plataformas via WebAPI.

Embora a plataforma tenha demonstrado bastante valor, há que ponderar até que ponto se consegue viver com as falhas que atualmente estão presentes, em especial na a integração do Visual Studio com um Mac OS, em que por vezes não se consegue iniciar o Agente que faz a compilação e o próprio simulador. Algumas falhas na iniciação do emulador Android, que pura e simplesmente bloqueiam o Visual Studio.

E temos de desenvolvimento, um pormenor interessante que gostei foi a similaridade no desenvolvimento de *App's* para Android com *Xamarin* comparativamente com o método tradicional, usando Java e Android Studio. Ou seja, todo o código é muito idêntico, desde a criação das *Views* (seja a nível do XML ou editor visual), *Activities* e *Adapters*. Tendo eu experiência profissional em .Net, e com os conhecimentos adquiridos em Sistemas Embebidos, foi relativamente fácil conseguir chegar ao resultado final.

A nível de desenvolvimento da *App* para iOS, não posso dizer o mesmo, já que não tive qualquer contacto com desenvolvimento nativo para esta plataforma, apenas me foi possível constatar que o ambiente visual é exatamente igual ao que se encontra no *Xcode*.

Após algum estudo sobre plataforma, a resposta para a similaridade deve-se a um facto, todas as funções da *framework* .Net têm um correspondente (chamado de *binding*) na

linguagem da plataforma a que se destina, no caso do Android será Java e o caso do iOS *Objective-C* (o *Swift* ainda não está presente nos *bindings* do *Xamarin*, pelo menos de forma oficial).

Quanto a dificuldades no projeto, a componente mobile foi claramente o grande desafio, e este teve um peso idêntico entre configurações e código necessário para atingir o objetivo. A nível de configurações, a incidência foi totalmente no emulador Android, já que este tem mais possibilidades de parametrização do que o emulador iOS, podendo ter inúmeras versões do Android, desde a 2.1 (não considerando outras versões já consideradas obsoletas) até à mais recente 7.0 (à data da realização do projeto), e ainda cada uma das versões têm a possibilidade de escolha várias imagens do sistema operativo, que divergem em versões x86, x64, versão standard, versão com Google *API's*. Para este projeto, a versão utilizada para permitir todas as funcionalidades foi a versão 5.0.1 de arquitetura x86 com Google *API's*. Sem as *API's* da Google, as *push notifications* não funcionam e sem uma imagem de arquitetura x86, não temos performance (através de aceleração de hardware), portanto é crucial escolher bem a imagem a utilizar.

Além da questão da imagem, também é necessário ter em conta algumas configurações do próprio projeto Android, segue uma imagem com as configurações obrigatórias para o cenário Mac OS com Xamarin Studio:

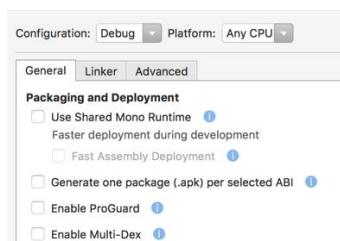


Figura 9 - Configuração do projeto Android - Separador "General"

Após desativar as bibliotecas partilhadas da *framework* Mono, é também preciso garantir que a arquitetura x86 está contemplada na criação do pacote de teste no emulador, dado que estamos a usar uma imagem x86.

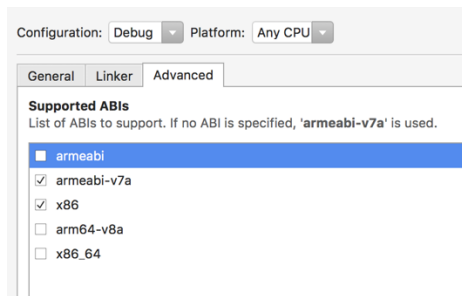
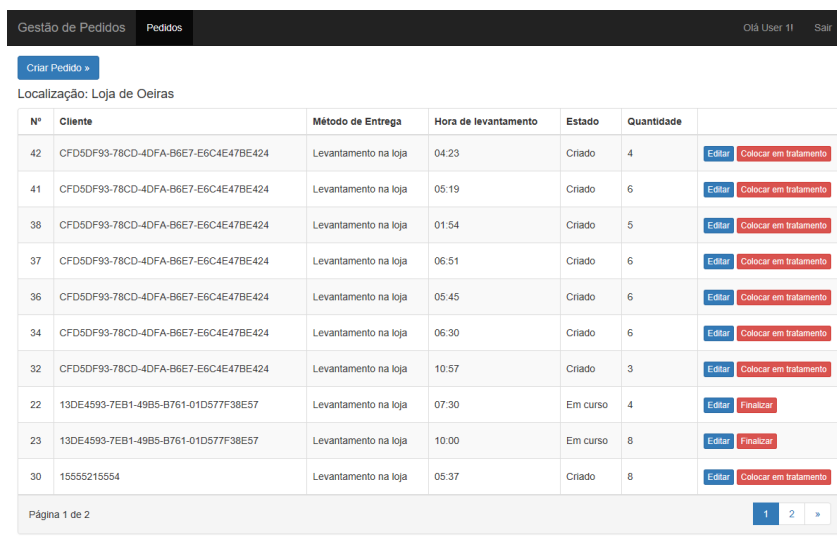


Figura 10 - Configuração do projeto Android - Separador “Advanced”

Quanto a dificuldades no código, esta incidiu na *App* para iOS, que como referi anteriormente não tinha qualquer *know-how* sobre o método de desenvolvimento para esta plataforma, pelo que foi necessária pesquisa. Um dos pormenores que me dei conta, foi que a pesquisa de resolução de problemas em Xamarin iOS, pode por vezes ser dolorosa, já que muitas vezes as pesquisas devolvem resultados direcionados a desenvolvimento nativo com *Objective-C/Swift*.

Seguem algumas imagens representativas da plataforma, tanto a área de *Backoffice* como Aplicação Cliente.



N°	Cliente	Método de Entrega	Hora de levantamento	Estado	Quantidade	
42	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	04:23	Criado	4	Editar Colocar em tratamento
41	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	05:19	Criado	6	Editar Colocar em tratamento
38	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	01:54	Criado	5	Editar Colocar em tratamento
37	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	06:51	Criado	6	Editar Colocar em tratamento
36	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	05:45	Criado	6	Editar Colocar em tratamento
34	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	06:30	Criado	6	Editar Colocar em tratamento
32	CFD5DF93-78CD-4DFA-B6E7-E6C4E47BE424	Levantamento na loja	10:57	Criado	3	Editar Colocar em tratamento
22	13DE4593-7EB1-49B5-B761-01D577F38E57	Levantamento na loja	07:30	Em curso	4	Editar Finalizar
23	13DE4593-7EB1-49B5-B761-01D577F38E57	Levantamento na loja	10:00	Em curso	8	Editar Finalizar
30	15555215554	Levantamento na loja	05:37	Criado	8	Editar Colocar em tratamento

Página 1 de 2

© 2017 - Plataforma de Gestão de Pedidos

Figura 11 - Vista disponibilizada a um utilizador/operador de loja

Este ecrã permite ver os pedidos que estão a entrar para a loja onde o utilizador está assignado, a lista é atualizada automaticamente, não sendo necessária intervenção manual do utilizador.

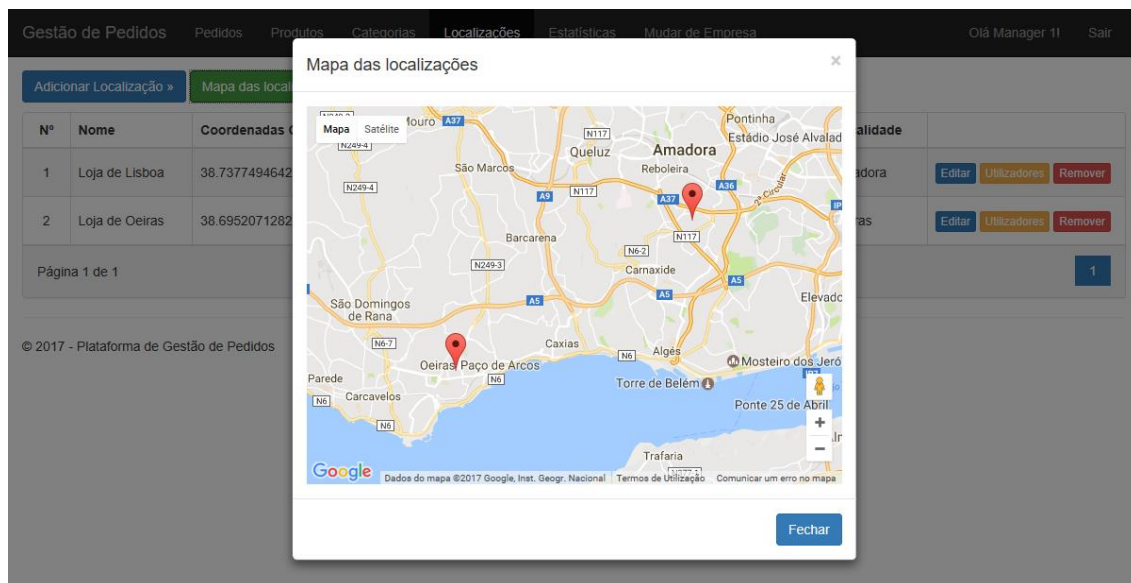


Figura 12 - Ecrã de visualização dos locais de uma empresa

Este ecrã tem uma particularidade interessante, que é a integração com a API de mapas da Google. Esta integração foi um pouco demorada, embora a API seja relativamente simples, dado que a *framework* de design usada, o Bootstrap, causa algumas dificuldades na visualização correta dos mapas. Desta forma, está disponibilizado no Anexo B, um guia para integração onde é explicado passo a passo o código necessário para chegar a um resultado idêntico ao que consta na imagem.

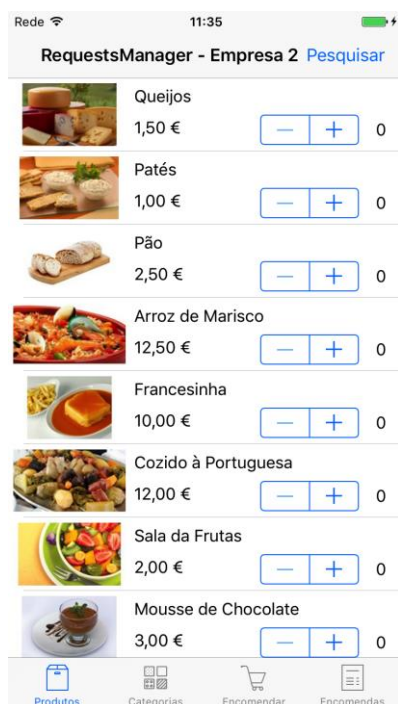


Figura 13 - Ecrã inicial da Aplicação móvel em iOS

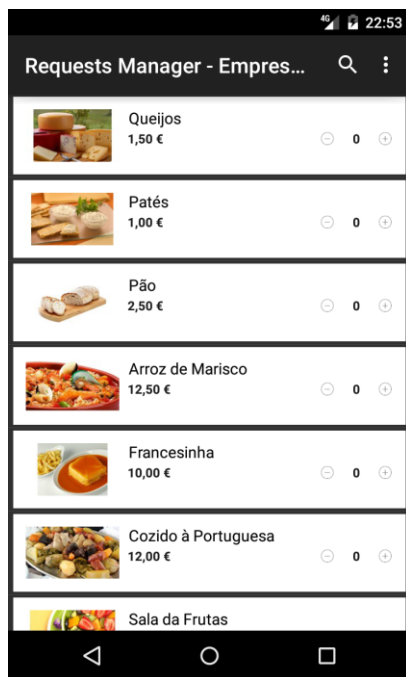


Figura 14 - Ecrã inicial da Aplicação móvel em Android

Como é possível verificar, visualmente, as aplicações têm um comportamento muito semelhante, no entanto seguem as práticas que cada sistema requer. A navegação no caso do Android é feita pelo menu no topo superior direito, já o caso do iOS é feito com uma barra de botões de navegação.

5 Conclusões e trabalho futuro

Este Trabalho Final de Curso trouxe uma importante abordagem inicial ao mundo mobile, tendo contribuído significativamente para o meu desenvolvimento académico e profissional. Foi possível com este trabalho ter uma melhor noção do *Cross Platform Development*, disponibilizado pela Microsoft, embora a nível de suporte não considere ainda num estado perfeito, diria que há potencial e consegue-se ganhos significativos no tempo de desenvolvimento.

Uma das questões que mais diferença senti e que provavelmente investiria mais se tivesse a noção que tenho neste momento, foi a forma de desenvolver as aplicações móveis e a própria API, com recurso ao paralelismo que a framework .Net disponibiliza. Dada a dependência que temos de sistemas externos, a possibilidade correr tarefas assíncronas é extremamente essencial para dar uma melhor experiência aos utilizadores. Apesar de ter introduzido estas técnicas na componente Mobile e API, penso que com mais conhecimentos, conseguiria ter um aproveitamento melhor neste campo.

Como possibilidade de implementações futuras, vejo a integração do sistema de *Push Notification* nos dispositivos iOS, a melhoria da página das estatísticas, a implementação de *business intelligence*, de forma a perceber os clientes e conseguir direcionar certas campanhas de produtos. Outros aspetos, poderiam ser a maior interação com os sensores que os equipamentos móveis disponibilizam, como a localização, Bluetooth (para *bacons*), conseguindo uma melhor experiência para o utilizador.

6 Bibliografia

WebAPI – <https://www.asp.net/web-api>

Xamarin – <https://developer.xamarin.com/>

7 Glossário

Xcode – Plataforma de desenvolvimento de aplicações para o ecossistema da Apple.

Objective-C/Swift – Linguagens de programação utilizadas para o desenvolvimento na plataforma da Apple.

MVC – Model View Controller, representa um padrão de arquitetura de sistemas maioritariamente utilizado para aplicações Web.

Cloud ou Cloud Computing – Plataformas que disponibilizam recursos, seja *hardware* ou *software*, tipicamente em infraestruturas baseadas em sistemas distribuídos.

Integração contínua (*continuous integration*) – É uma prática utilizada no desenvolvimento de software, que permite validar o estado atual do código fonte que está publicado nos repositórios.

Push Notifications – Tecnologia utilizada para envio de mensagens para os equipamentos cliente. Neste cenário é o servidor que toma a iniciativa do envio da mensagem.

ANEXO A

Requisitos

Requisitos funcionais

ID	Requisito	Sistema	Perfil/Perfis
1	O sistema deve permitir o registo de empresas ou de grupos de empresas.	Backoffice	Admin
2	O sistema deve permitir vários utilizadores.	Backoffice	Admin
3	Um utilizador pode pertencer a várias empresas	Backoffice	Admin
4	Um utilizador é constituído por um Nome de Utilizador, Password e E-mail.	Backoffice	N/A
5	Um produto é constituído por: Nome, Descrição, Categoria, Preço, Tempo de Preparação, e Imagem.	Backoffice	N/A
6	Um pedido é constituído por: Identificação do Cliente, Produtos, Método de Entrega, Local de Levantamento, Data do Pedido, Hora para Levantamento, Estado, IP Cliente, Sistema Operativo.	Backoffice	N/A
7	Um local de levantamento é constituído pelos elementos essenciais de uma morada e das coordenadas GPS.	Backoffice	N/A
8	Deverá ser possível ver a lista de pedidos atualizada em tempo real, pela ordem de hora de levantamento e data do pedido.	Backoffice	Operador, Manager
9	Deverá ser possível alterar o estado de um pedido para iniciado ou concluído.	Backoffice	Operador
10	Sempre que um pedido não seja dado como concluído após 2 horas (depois de ter sido dado como iniciado), este terá de passar para expirado.	Backoffice	N/D
11	Deverá ser possível ver a estatística de pedidos diários.	Backoffice	Manager
12	Deverá ser possível adicionar/alterar/arquivar produtos.	Backoffice	Manager
13	A aplicação deverá mostrar uma lista dos produtos disponíveis para efetuar pedido.	Mobile App Demo	Utilizador
14	O utilizador deverá conseguir selecionar vários desses produtos, e ter a indicação do custo total.	Mobile App Demo	Utilizador
15	O utilizador deverá conseguir efetuar o pedido, após selecionar os produtos e de os confirmar. Será indicado um número de pedido ao cliente.	Mobile App Demo	Utilizador
16	A aplicação deverá registar como identificação o número de telemóvel do utilizador (se não for possível, por questões de segurança, utilizador o Hardware ID do equipamento).	Mobile App Demo	N/D
17	A aplicação deverá receber um "push notification" assim que um pedido for dado como concluído.	Mobile App Demo	N/D

Requisitos não funcionais

ID	Requisito	Sistema
1	O <i>backoffice</i> deverá ser compatível com os browsers mais recentes: IE9+, Chrome, Firefox e Opera.	<i>Backoffice</i>
2	O <i>backoffice</i> deverá funcionar como SPA (Single Page Application), tendo interações apenas via AJAX.	<i>Backoffice</i>
3	Para tratamento de dados específicos, terão de ser apresentados popups com os formulários, de forma a focar o utilizador nesses mesmo dados.	<i>Backoffice</i>
4	Toda a interação deverá ser possível apenas com teclado, com recurso a atalhos.	<i>Backoffice</i>
5	De 3 em 3 segundos deverá ser feito um refrescamento da lista de pedidos, aproximando ao máximo de um sistema em tempo real.	<i>Backoffice</i>
6	O <i>backoffice</i> deverá ser alojado na plataforma Azure, de forma a permitir uma disponibilidade de 99,95%.	<i>Backoffice</i>
7	A App Mobile deverá correr em sistemas Android de versão igual ou superior a 4.4 e sistemas iOS versão 9.X.	Mobile App Demo
8	Para correr a App Mobile serão necessários também os seguintes requisitos mínimos de hardware: 512MB de memória RAM e um processador.	Mobile App Demo
10	As <i>push notifications</i> devem ser implementadas.	Mobile App Demo

ANEXO B

Integração com Google Maps (guia para programadores)

Este guia tem como objetivo mostrar, passo a passo, como integrar a API do Google Maps num website. Esta API foi usada no projeto para as localizações das lojas. Como objetivo deste guia, iremos ter então dois pontos marcados no Mapa, um será a Universidade Lusófona no Campo Grande (38.758108, -9.152188) e outro o Aeroporto de Lisboa (38.7697127, -9.12845).

O primeiro passo a efetuar para integrar o Google Maps é obter a key necessária para ter acesso à API, para tal é necessário no portal de *developers* registar um projeto e obter a chave para a API do google maps. À data, o portal está acessível através do endereço <http://console.developers.google.com/>.

Depois de se obter a chave, temos de colocar a seguinte tag HTML no site:

```
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&key=<key>"
async defer></script>
```

Nota: substituir o valor <key> pela chave obtida no portal.

De seguida teremos então de implementar a chamada à API, que passa por instanciar um objeto *Map*, que irá ter como referência uma *div*, onde se prende renderizar o mapa. Esta *div* não tem nenhuma característica especial, apenas deve ser especificado o seu tamanho (através do atributo *style* ou através de classe CSS):

```
<div id="map-canvas" style="width:500px; height:400px;"></div>

var map = new google.maps.Map(document.getElementById("map-canvas"), {
  mapTypeId: google.maps.MapTypeId.ROADMAP
});
```

Depois de termos a instância da *div* para o Mapa, teremos de adicionar os pontos do mapa, para tal, precisamos de um objeto do tipo *google.maps.LatLngBounds*, que vai conter dois elementos *google.maps.Marker*:

```
var bounds = new google.maps.LatLngBounds();

bounds.extend((new google.maps.Marker({
  position: new google.maps.LatLng(38.758108, -9.152188),
  map: map
})).getPosition());
bounds.extend((new google.maps.Marker({
  position: new google.maps.LatLng(38.7697127, -9.12845),
  map: map
})).getPosition());
```

Depois dos pontos adicionados, só precisamos então de garantir que o mapa fica centrado nos pontos, de forma a conseguir visualizar os dois pontos. É recomendado colocar esta ação num *listener* de redimensionamento da janela (*resize listener*):

```
map.setCenter(bounds.getCenter());
map.fitBounds(bounds);
```

De forma a resolver algumas dúvidas sobre como colocar cada um dos blocos de código, fica um exemplo mais completo:

```
<script>
  var map = new google.maps.Map(document.getElementById("map-canvas"), {
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });
  var geocoder = new google.maps.Geocoder;
  var bounds = new google.maps.LatLngBounds();

  function initialize() {
    bounds.extend((new google.maps.Marker({
      position: new google.maps.LatLng(38.758108, -9.152188),
      map: map
    })).getPosition());
    bounds.extend((new google.maps.Marker({
      position: new google.maps.LatLng(38.7697127, -9.12845),
      map: map
    })).getPosition());
  };
  google.maps.event.addDomListener(window, "resize", resizingMap());

  function resizingMap() {
    if (typeof map == "undefined") return;
    google.maps.event.trigger(map, "resize");
    if(bounds != null) {
      map.setCenter(bounds.getCenter());
      map.fitBounds(bounds);
    }
  }

  $(function () {
    initialize();
    setTimeout(function () { resizingMap(); }, 400);
  });
</script>
```

Após a conclusão do guia, o resultado expectável ao nível do mapa é o seguinte:

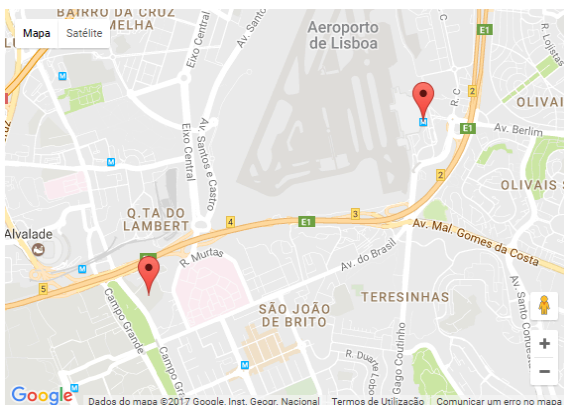


Figura 15 - Resultado integração API Google Maps

A documentação da API Google Maps está disponível para consulta no seguinte endereço: <https://developers.google.com/maps/documentation/javascript/>.