Projeto

Mestrado em Engenharia Informática – Computação Móvel

SDCU - Sistema de Divulgação de Comércio Urbano

António Miguel Pereira Ribeiro

Leiria, Novembro de 2012

Projeto

Mestrado em Engenharia Informática – Computação Móvel

SDCU - Sistema de Divulgação de Comércio Urbano

António Miguel Pereira Ribeiro

Projeto de Mestrado realizado sob a orientação da Doutora Catarina Helena Branco Simões da Silva e do Doutor Luís Filipe Fernandes Silva Marcelino , ambos Professores da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, Novembro de 2012

Agradecimentos

Aos Professores orientadores pelo apoio prestado na elaboração de todo o projeto.

À namorada e à irmã por serem uma peça fundamental no meu equilíbrio durante este projeto.

Aos pais pela paciência nos momentos mais difíceis.

Aos *testers* pela disponibilidade.

Resumo

As lojas de pequeno retalho estão cada vez mais em perigo extinção devido à constante expansão e crescimento das grandes superfícies. Esta expansão é feita nas mais variadas áreas de negócio, o que acaba por sufocar os mercados e leva ao encerramento ou mesmo falência das lojas mais pequenas.

Com este projeto pretende-se proporcionar uma nova experiência de compras, permitindo aos consumidores pouparem dinheiro ao usufruírem de promoções e aos lojistas gerarem mais receitas. Para isso, é proposta uma arquitetura que suporte um sistema distribuído com a finalidade de permitir aos lojistas fazerem toda a gestão das suas lojas e promoções e aos consumidores acederem às lojas existentes e resgatarem as promoções. Este sistema permite fazer a integração com aplicações externas de forma a possibilitar a expansão desta aplicação.
No âmbito deste projeto foi desenvolvida a parte *web* do sistema para que esta possa ser construída de forma sólida e estudada a aceitação que terá no mercado. A parte *web* é constituída por um portal onde os lojistas executam as suas tarefas de gestão e os serviços REST para disponibilizar os dados a terceiros que tenham o desejo de construir aplicações que interajam com esta.

Após o desenvolvimento, a aplicação foi testada por indivíduos ligados ao comércio de pequeno retalho. A familiaridade dos *testers* com o pequeno comércio permitiu obter um *feedback* real do ponto de vista do comerciante. Se estes testes tivessem sido executados por pessoas alheias a este tipo de negócios, o resultado destes poderia ser enganador uma vez que apenas conhecem o pequeno retalho do ponto de vista do consumidor. O *feedback* obtido dá conta de possíveis barreiras à implementação do sistema no mercado, mas fornece uma perspetiva muito mais positiva, face às negativas, em que ajudará os pequenos negócios a expandirem-se e atrair novos clientes.
Palavras-chave: web, comércio, promoção

Abstract

Small businesses stores are endangered due to the expansion and growth of large retail stores. This expansion is performed in various areas of business, which ultimately suffocate markets and leads to foreclosure or even bankruptcy of small business stores.

This project aims to provide a new shopping experience and allow consumers to save money by taking advantage of promotions while retailers generate more revenue. In order to achieve this, it is proposed an architecture that supports a distributed system with the goal of allowing merchants to make all management of its stores and promotions and allowing consumers to access the existing stores and redeem promotions. This system allows the integration with external applications in order to enable the expansion of this application.
As part of the project, the web tier was developed so it can be built on a solid basis and studied the acceptance that this may have in the market. This tier is composed by a web portal, where merchants perform their management tasks, and by a REST service to provide the system's data to third party developers that are willing to build applications to interact with this one.

After the development, the application was tested by individuals that are familiar with the small businesses. This connection between testers and small businesses allowed to get a real feedback from the merchant point of view. If the tests were performed by people who were not related to this kind of business, then the test results could be misleading since they only know small retails stores from a consumer point of view. The feedback obtained realizes potential barriers to implementation of the system on the market, but provides a much more positive perspective, given the negative, which will help small businesses expand and attract new customers.

Key-Words: web, shopping, promotion

Índice de Figuras

Índice de Tabelas

Lista de Siglas

ACID	Atomicity, Consistency, Isolation, Durability
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ATM	Automated Teller Machine
CRUD	Create, Retrieve, Update e Delete
DER	Diagrama Entidade-Relacionamento
DOM	Document Object Model
DQL	Doctrine Query Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JavaME	Java Micro Edition
JSON	JavaScript Object Notation
MVC	Model View Controller
ORM	Object Relational Mapping
PHP	PHP: Hypertext Preprocessor
POS	Point Of Sale
REST	Representational State Transfer
RFID	Radio-Frequency IDentification
SAD	Sistemas de Apoio à Decisão
SDCU	Sistema de Divulgação de Comércio Urbano
SDK	Software Development SDK
SGBD	Sistema de Gestão de Base de Dados
SMS	Short Message Service
SO	Sistema Operativo
SOAP	Simple Object Access Protocol
SQL	Structured Query Language

URL	Uniform Resource Locator
WP7	Windows Phone 7
WSDL	Web Services Description Language
XML	eXtensible Markup Language

Índice

1 Introdução

Nos dias que correm, um grande desafio que se tem colocado às famílias portuguesas tem sido a gestão económica familiar. A crise que percorre o mundo tem levado a atrasos no pagamento de salários, cortes nos mesmos e em casos extremos a despedimentos. Desta forma, é natural que as pessoas sejam atraídas por produtos a preços mais baixos, muitas vezes de marcas menos conhecidas, vulgarmente designados de produtos de marca branca, de forma a que consigam comprar mais por menos dinheiro. Esta situação leva a que as lojas de pequeno retalho sintam dificuldades pelo facto de não possuírem marca própria, não conseguindo igualar os preços das grandes superfícies. Assim, é natural que estas lojas tendam a desaparecer face à concorrência das grandes superfícies comerciais e às promoções e ofertas por elas praticadas. Estas lojas dependem cada vez mais da lealdade dos clientes, seja pelo hábito, pelo atendimento mais personalizado, pela qualidade dos produtos ou apenas pela conveniência de estar mais perto de casa. Desta forma, é importante que as lojas mais pequenas se adaptem ao meio e que consigam elas lançar as suas próprias promoções de forma a criar alguma lealdade aos consumidores para garantir a sua existência.

1.1 Objetivos

Com este projeto pretende-se apresentar e construir uma solução *web* que permita aos lojistas fazerem a divulgação das promoções que praticam nas suas lojas. O portal *web* desenvolvido será utilizado pelos lojistas para poderem fazer a gestão das suas lojas e promoções que decorrem nas mesmas e comunicar diretamente com os consumidores. Os serviços REST permitirão fornecer os dados presentes no sistema a terceiros para que estes possam desenvolver aplicações direcionadas ao consumidor com a finalidade deste aceder às promoções e fazer uso das mesmas. Com estas ferramentas a finalidade é que os consumidores consigam poupar dinheiro nas suas compras através das promoções ao mesmo

tempo que os lojistas conseguem gerar mais receitas, fidelizar e angariar novos clientes para que o seu negócio se mantenha financeiramente saudável. Para que os custos inerentes ao funcionamento da plataforma sejam os mais reduzidos possíveis, e assim cobrar o mínimo aos lojistas, todas as tecnologias, plataformas e ferramentas utilizadas deverão ser gratuitas.

1.2 Requisitos

Os requisitos para esta plataforma apresentam aquilo que o sistema deve permitir executar (requisitos funcionais) e as caracteristicas que o sistema deve possuir (requisitos não funcionais).

1.2.1 Requisitos funcionais

Os requisitos funcionais foram divididos em duas secções. Uma descreve as funcionalidades que deverão estar disponíveis no portal *web* para uso por parte do utilizador. A segunda descreve as funcionalidades disponibilizadas através do *webservice*.

1.2.1.1 Portal *web*

Para que a aplicação permita aos utilizadores gerirem as suas lojas, é necessário em primeiro lugar haver uma gestão de utilizadores. Estes deverão puder autenticar-se na plataforma, através de um nome de utilizador e palavra-passe e, em função das permissões associadas poderá executar diferentes tarefas. Esta gestão de utilizadores só poderá ser efetuada por aqueles que possuam a permissão para tal e inclui a adição, remoção, edição e bloqueio dos utilizadores.

Uma vez que o foco da plataforma é a divulgação de promoções existentes nas lojas, é necessário haver uma gestão das lojas presentes no sistema. Apenas os utilizadores com permissões para tal podem adicionar, remover e editar as lojas.

Um utilizador pode ser associado a várias lojas. Um utilizador pode ser associado como gestor e/ou lojista. Um gestor pode editar os detalhes da loja, moderar as suas promoções e adicionar outros gestores e lojistas; o seu principal objetivo é garantir que as promoções

inseridas para as lojas que gere são lícitas e válidas de forma a que o sistema se mantenha livre de dados indesejados. Um utilizador associado como lojista pode, se tiver as permissões corretas, criar promoções, confirmar o resgate de uma promoção, contactar com os consumidores e editar detalhes básicos da loja, como o caso da morada e dos contactos.

A promoções que serão divulgadas aos consumidores deverão ser inseridas pelos lojistas. Estas promoções deverão ter um número máximo de resgates confirmados. Um resgate é a ação do consumidor obter um código para a promoção que lhe dará acesso à mesma. Esta limitação permitirá controlar abusos no aproveitamento das promoções, sendo que um consumidor pode resgatar uma promoção e utilizar um código uma vez. O lojista deverá ser possibilitado de confirmar o código apresentado pelo consumidor aquando do decorrer do processo de compra. Quando uma promoção é criada, os consumidores podem resgatar códigos até o número máximo de resgates, caso exista, seja atingido, a data limite seja atingida ou até que um lojista cancele a promoção. Aquando desta ação deverá ser possível informar a razão de cancelamento. Uma promoção não deverá ser editável para que não seja possível alterar as datas ou descrição das mesmas, de forma a não suscitar suspeitas de manipulação por parte dos consumidores.

Às promoções e lojas deverá ser possível adicionar produtos. Estes produtos deverão ser partilhados por todas as lojas de forma a não serem replicados registos na base de dados. Estas associações permitiram pesquisas com foco em algum produto que o consumidor deseje adquirir.

1.2.1.2 Webservice

Deverá existir um *webservice* que permita a terceiros construirem aplicações dirigidas ao consumidor final e que permita a estes interagirem com o sistema. Este *webservice* deverá suportar mecanismos de segurança que não permita a um utilizador impersonar outro. Todas as ações que um utilizador execute devem apenas afetar o próprio e manter os dados alheios a ele intactos.

O *webservice* deverá permitir um utilizador autenticar-se através do seu nome de utilizador e palavra-passe.

O *webservice* deverá permitir navegar pelo diretório geográfico, filtrando então as lojas por localidade. Deverá também permitir filtrar as lojas pelo seu tipo.

Deverá estar disponível uma funcionalidade de pesquisa através das palavras-chave que constituem cada promoção. A resposta a este pedido deverá ser constituído pelas promoções cujas palavras-chave contenham o termo de pesquisa do utilizador.

Deverá ser possível obter os detalhes de uma loja, tais como o nome, contactos, morada e promoções em vigor.

O *webservice* deverá também permitir ao consumidor resgatar uma promoção, fornecendo então um código que servirá para o lojista confirmar no ato da compra. Também deverão estar disponíveis os códigos das promoções previamente resgatadas para referência do utilizador.

A troca de mensagens deverá ser possível entre o consumidor e os lojistas. Este serviço de mensagens deverá permitir uma troca de mensagens em tempo real de forma a agilizar a comunicação entre os dois pontos.

1.2.2 Requisitos não funcionais

O sistema deverá ser seguro e viável de forma a que os dados fornecidos pelos utilizadores estejam protegidos contra ataques. É igualmente importante manter a disponibilidade do sistema para que o utilizador tenha uma experiência fluída durante a utilização da plataforma.

Para que não sejam incumbidos encargos aos lojistas pelos custos da plataforma, esta deverá ser desenvolvida sobre tecnologias gratuitas. Assim, é possível fornecer aos lojistas uma plataforma de baixo custo, ou mesmo gratuita, com a finalidade de divulgar a sua loja e promoções.

1.3 Metodologia

O desenvolvimento deste projeto deverá faseado para que a coerência deste não se perca ao longo de todo o processo. Inicialmente é necessário identificar os requisitos da plataforma para que sejam conhecidas as funcionalidades que esta deve proporcionar aos utilizadores (requisitos funcionais) e outras características de cariz não funcional (requisitos não funcionais).

O segundo passo é o levantamento de estado da arte que permite conhecer que estudos já foram feitos e que plataformas semelhantes existem no mercado. Este levantamento permite identificar quais as lacunas que os outros sistemas têm e que podem ser colmatadas pelo apresentado neste projeto.

No terceiro passo deve ser proposta uma arquitetura e selecionar quais as tecnologias e *frameworks* que servirão no desenvolvimento da plataforma. Esta seleção é importante uma vez que devem ser escolhidas tecnologias robustas que permitam atingir os requisitos do sistema.

O desenvolvimento é elaborado na quarta fase em que são implementados os requisitos identificados anteriormente com recurso às tecnologias selecionadas.

Depois da construção deste sistema, é necessário planear os testes a efetuar. Estes devem ser executados por pessoas que estejam relacionadas com a área do pequeno comércio para que estes sejam o mais realistas possível. Deve existir um *briefing* inicial a cada utilizador sobre o propósito e funcionamento do sistema e, aquando do início dos testes, deve ser oferecido um guião com a lista de tarefas a concluir. Este guião deve ser igual para todos os utilizadores, sendo tidas em consideração métricas como o tempo que demorou a até ser concluída determinada tarefa, se a tarefa foi ou não concluída e quantos erros foram cometidos até o utilizador atingir o objetivo de cada uma ou declarar "desistência". Estes dados poderão indicar quais os aspetos que podem ser melhorados, recolher *feedback* dos utilizadores e obter uma primeira impressão sobre a viabilidade que os lojistas vêem no sistema e de que forma podem os folhetos em papel serem substituídos pelos digitais.

1.4 Estrutura do documento

Neste relatório é descrito todo o processo de desenvolvimento do produto desde a análise de trabalhos já efetuados na área até às conclusões que resultaram do projeto. Depois desta

introdução é apresentado o estado da arte no Capítulo 2. Seguidamente, no Capítulo 3, é apresentada a arquitetura do sistema desenvolvido, as tecnologias que o compõe e as *frameworks* utilizadas na construção da plataforma. As funcionalidades implementadas são descritas no Capítulo 4. Os testes efetuados e resultados obtidos apresentados no Capítulo 5.

2 Estado da arte

Sendo o tópico de gestão orçamental familiar um tema já muito debatido em todo o mundo, é natural que já tenha sido alvo de estudos e que tenham sido feitas aplicações que auxiliem os consumidores na hora de fazer compras. Neste capítulo apresentamos o estado da arte dos sistemas que permitem aos lojistas publicitar as suas lojas, os produtos que comercializam e as promoções que praticam. As seguintes secções apresentam trabalhos científicos levados a cabo como as plataformas disponíveis dentro da área.

2.1 Aplicações não comerciais

Como já foi referido, existem muitos trabalhos que se debruçam sobre a temática da gestão familiar do consumo, recorrendo muitas vezes a tecnologia móvel. Heijden et al. [1] estudam a forma como os consumidores escolhem determinado produto e consideram várias alternativas quanto à forma de os ajudar nessa escolha utilizando tecnologia. Através de um dispositivo móvel, como um telemóvel, o utilizador é avisado de um potencial bom negócio ou de um produto do seu interesse que esteja a um bom preço. Este aviso pode ser feito de várias formas: por som, vibração e visão. Através de toques mais ou menos audíveis, uma maior ou menor intensidade de vibração (ou diferentes padrões) ou informação disponibilizada no visor, o consumidor apercebia-se da qualidade do negócio proposto para o utilizador. Para a aplicação saber qual o produto a ser analisado, a sua informação poderia ser obtida de várias formas, como por exemplo, código de barras ou RFID. Em ambos os casos, a quantidade de informação é limitada, pelo que é necessário um servidor central onde são armazenados os atributos a serem avaliados pelo *software*.

Por outro lado Kowatsch & Maass [2] propõem um sistema que funciona dentro do local de consumo, com o objetivo de apresentar, na própria loja, não só as informações dos produtos mas também de permitir ao utilizador pedir informações tais como "quais os acessórios disponíveis?" ou "quais os produtos alternativos?". Para que isto funcione, é necessário que os produtos tenham um código de barras, RFID ou qualquer outra forma de dar entrada dos dados no dispositivo móvel.

Em 2007, Westermanetal et. al [3], estudaram a utilização de sistemas de apoio à decisão (SADs) neste contexto para avaliar se seria possível influenciar os clientes de forma positiva.

Nos testes efetuados verificou-se que os clientes faziam escolhas mais rapidamente sem recurso a nenhum SAD e faziam-no com base no seu conhecimento empírico. Por outro lado, os que utilizavam somente o SAD tomavam decisões um pouco mais lentamente mas, ao seguirem a sua recomendação, mostravam-se mais satisfeitos do que aqueles que não consideravam a proposta do SAD. Por último, a mistura da utilização do SAD com a presença do produto, revelou-se a experiência mais lenta. Esta diferença dever-se-á ao cruzamento da informação fornecida pelo SAD com o conhecimento empírico, o que levará a análises mais minuciosas e cuidadas mas que deixará o cliente mais satisfeito e confiante da sua compra.
Em [4] foi estudado o efeito da estética em lojas *online*. Estes mediram dois fatores durante o estudo: processo de compra e gozo de compra. O primeiro mede a poupança de tempo quando se passa pelo processo de compra e o segundo mede o gozo e a experiência durante o processo de compra. Para se conseguirem medir estes fatores, foram postos à prova dois tipos de interface: uma clássica e uma expressiva. A clássica identifica-se bastante com as lojas *online* atuais, isto é, uma lista de produtos, catálogo, cores escuras sobre um fundo branco. A expressiva conta com uma palete de cores mais escuras, com menos contraste e com um grande dinamismo na página. Em conclusão, percebeu-se que quando os utilizadores estão orientados para fazer determinada compra, a interface clássica permite que estes tenham um melhor desempenho no primeiro fator (processo de compra) mas o segundo fator (gozo de compra) torna-se irrelevante. Por outro lado, quando um utilizador navega pela loja em busca de algo que suscite interesse, a interface expressiva aumenta o interesse do utilizador uma vez que a tarefa não se torna monótona e comum aos outros *websites*. No entanto, quando este encontra algo do seu interesse, prefere que a interface se identifique com a clássica para que o processo de compra seja acelerado.

2.2 Plataformas comerciais

A Amazon[1] fornece um serviço de vendas para outros profissionais. Para que outros retalhistas possam vender a partir desta plataforma é necessário que se registem e insiram os produtos através de um *backoffice* para o efeito. Este serviço tem um custo mensal acrescido de um custo por transação efetuada. Os pagamentos dos consumidores é tratado pela Amazon e é esta que entrega o valor pago pelo consumidor (deduzido das taxas da transação) ao vendedor. Alternativamente, também é possível delegar a responsabilidade de entrega dos

[1] http://www.amazonservices.com/selling/benefits.htm

produtos à Amazon. Os produtos presentes nesta loja assim como os inseridos por terceiros, são listados na aplicação móvel da Amazon disponível para iOS e Android.

A Google lançou o serviço "Merchant center"[2] que permite aos comerciantes colocarem os seus produtos na base de conhecimento do Google para que este devolva os produtos em pesquisas na sua aplicação "Shopper" ou mesmo no Google Search. A população dos produtos pode ser feita via *backoffice*, através de *feeds* (ficheiro XML ou dividido por *tabs*) ou integração com outras lojas *online* (eBay por exemplo). Este serviço não tem qualquer custo mas apenas está disponível para os Estados Unidos da América.

O "ShopSavvy"[3] é uma plataforma que permite aos retalhistas publicarem os seus produtos de forma gratuita através de um ficheiro num formato específico que será interpretado pelo sistema. O serviço base, que contém apenas a publicação dos artigos, não tem qualquer custo e fica à distância de um registo. Caso o comerciante esteja interessado em publicidade por parte da "ShopSavvy", estes têm planos para tal. Os produtos constantes nesta base de dados podem ser consultados através de uma aplicação disponível para Android, iOS e Windows Phone 7.

No modelo semelhante à aplicação anterior, a "KuantoKusta"[4] (proprietária do mesmo site) permite fazer pesquisas em lojas de retalho em Portugal mas as suas categorias de indexação são focadas em produtos que não essenciais, tais como equipamentos tecnológicos, livros e decoração. Os retalhistas podem ver a sua loja presente neste sistema através de um registo, com pagamento mensal, e disponibilização de ficheiros com várias informações sobre a loja, produtos e preços.

A aplicação "Milo Local Shopping"[5] permite os lojistas terem a sua loja identificada num mapa e apresentar os produtos que oferece. Para que este sistema tenha conhecimento da loja e dos produtos é necessário fornecer os dados. Estes dados podem ser facultados através de *feed*, API ou diretamente do POS. Para esta última alternativa funcionar, é necessário que o lojista utilize um POS compatível com a aplicação "Fetch". Esta aplicação irá recolher a informação do POS periodicamente e enviá-la para o sistema "Milo".

[2]http://www.google.com/merchants

[3]http://shopsavvy.com/

[4]http://www.kuantokusta.pt/mobile

[5]http://milo.com/retailers/get-started.html

2.3 Conclusão

Neste capítulo é apresentada uma panorâmica geral dos trabalhos e sistemas que disponibilizam aos comerciantes várias alternativas no âmbito da divulgação da loja e dos seus produtos e apoio ao consumidor nas compras.

A Tabela 1 apresenta uma comparação entre as diferentes aplicações comerciais apresentadas na secção 2.2.

Aplicação	Encargos Fixos	Associação de produtos	Lançamento de promoções	População de dados
Amazon	Sim	Sim	Sim	Via *backoffice*
Google Merchant Center	Não	Sim	Não	Via *backoffice* *Feed* Integração
ShopSavvy	Não	Sim	Não	*Feed*
KuantoKusta	Sim	Sim	Não	*Feed*
Milo Local Shopping	Sim	Sim	Não	*Feed* Integração

Tabela 1 - Comparação das aplicações comerciais analisadas

Analisando a tabela anterior, nenhum dos sistemas sem encargos fixos contém a funcionalidade de lançamento de promoções. À exceção da Amazon, os serviços indexam os produtos comercializados pela loja em questão e estes ficam numa base de conhecimento que muitas vezes servirá, por parte dos consumidores, para comparação de preços.
Com o sistema a ser desenvolvido pretende-se que se ofereça aos comerciantes um sistema sem encargos fixos, no qual se possam associar os produtos que comercializa e fazer o lançamento de promoções. O carregamento de informação para a plataforma deverá ser feito via *backoffice*. Apesar de puder ser um pouco mais demorado que um processo automático, a inserção através de *backoffice* é *user-friendly* e dispensa de *software* para gerar um ficheiro que faça o papel de *feed* para a plataforma.

3 Arquitetura, tecnologias e *frameworks*

Neste capítulo apresentamos uma proposta da arquitetura para o sistema a ser desenvolvido seguida das tecnologias e *frameworks* que serão parte integrante desta arquitetura.

3.1 Arquitetura proposta

Para ir de encontro ao objetivo deste trabalho é necessário construir um *backoffice* em que os lojistas poderão gerir as respetivas lojas e promoções que decorrem nas mesmas. Uma vez que a grande maioria dos lojistas de pequeno retalho são pessoas que não têm um particular interesse pela tecnologia ou que a utilizam para fins básicos e de forma tradicional, como por exemplo a navegação na internet através do navegador utilizando um rato e um teclado, foi construído de um portal *web* que colocará o utilizador neste ambiente "familiar" da internet. Ao colocar-se o utilizador num ambiente que já tem alguma familiaridade, como é o caso do navegador, fará com que ele se sinta mais confortável e confiante na utilização. Se fosse utilizada uma aplicação *desktop* ou nativa de alguma plataforma móvel, o utilizador poderia sentir-se inibido e desconfortável principalmente se nunca tivesse tido contacto com estas plataformas.

Os *webservices* permitirão que terceiros possam desenvolver aplicações que integrem este sistema e que permitam aos consumidores finais interagirem com este sistema. Estes serviços permitirão efetuar operações como, por exemplo, registo de utilizadores, navegação pelo diretório de lojas ou resgate de promoções. A Figura 1 apresenta a arquitetura de alto nível proposta:

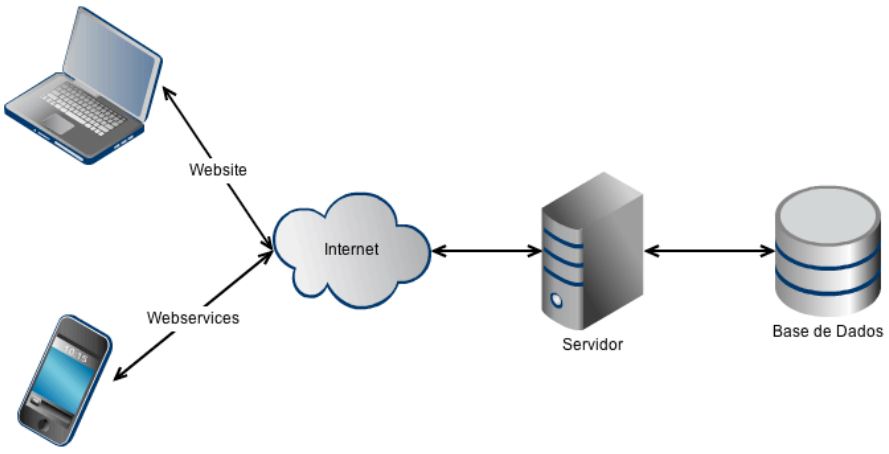

Figura 1 - Arquitetura de alto nível

Quando uma aplicação faz um pedido, este é enviado para o servidor e processado. O pedido é tratado no servidor e, dependendo da ação a realizar, pode ser feita, ou não, uma ligação à base de dados para obter ou manipular os dados que nela constam.
A Figura 2 ilustra a arquitetura interna do servidor e mostra a divisão das camadas existentes.

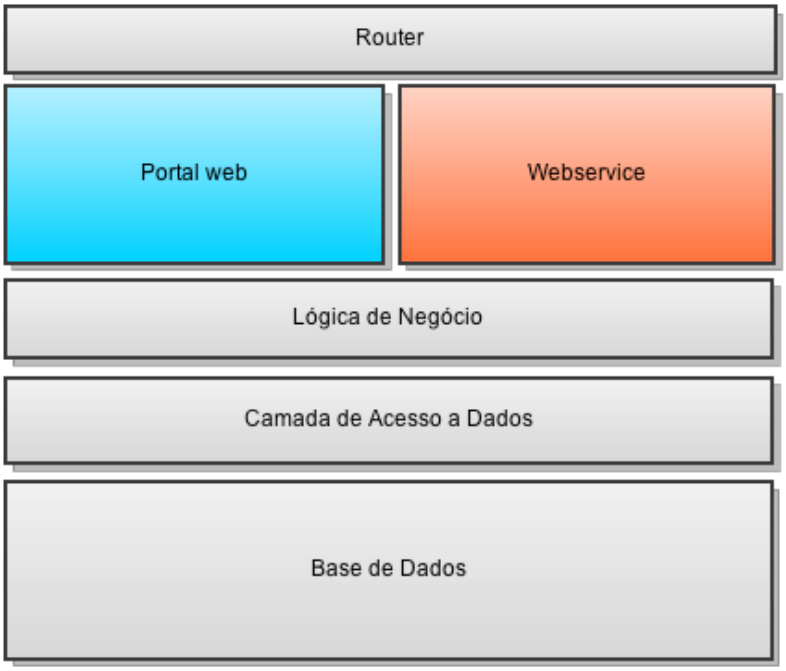

Figura 2 - Arquitetura interna do servidor

Quando um pedido é recebido a primeira camada, *Router*, verifica se o pedido é direcionado para o portal *web* ou para o *webservice* e redireciona-o internamente para a camada correspondente. Nesta o pedido é tratado consoante a ação a executar. Os dados que são enviados neste pedido são tratados e enviados para a camada de lógica de negócio. Esta recebe os dados uniformizados da camada do portal *web* e do *webservice*. Nesta camada é feita toda a lógica de negócio, como por exemplo, verificação da validade dos dados a serem guardados na base de dados. Os dados enviados para a camada de acesso a dados já se encontram tratados e estão prontos a serem inseridos na base de dados. No caso de se tratar da obtenção de dados, é feita uma pesquisa pelos dados na camada de acesso a dados à base de dados e estes serão devolvidos à lógica de negócio que, poderá ou não fazer algum pré-processamento, os devolverá para a camada de origem do pedido.

3.2 Tecnologias propostas

3.2.1 Web

O desenvolvimento do módulo *Web* deve contar com uma interface gráfica intuitiva mas também com um desempenho que agrade ao utilizador. Respeitando estes dois pontos, os utilizadores que fizerem uso da aplicação terão uma experiência agradável, pouco frustrante e que permitirá atrair novos utilizadores.
O PHP[6] é uma linguagem interpretada destinada ao desenvolvimento de aplicações *Web*. Esta é uma linguagem largamente utilizada na construção de *websites*, sejam eles de caráter pessoal ou comercial. A sua larga utilização levou a que o PHP se tornasse uma das linguagens mais populares e que o negócio de alojamento *web* tenha como oferta constante, e em preços competitivos, serviços de PHP. Esta linguagem, por ser interpretada e não compilada, permite que se corrijam *bugs* ou que se façam outras alterações *on-the-fly*, isto é, editando diretamente o ficheiro no servidor onde está instalado sem ser necessário compilar o código ou ter qualquer outro tipo de preparação pré ou pós alteração.

3.2.2 Webservice

A utilização de *webservices* é a forma mais comum de trocar informação entre sistemas diferentes que podem estar implementados em tecnologias diferentes, geograficamente separados e até mesmo desenvolvidos por entidades diferentes. Existem vários tipos de *webservices*, sendo os mais conhecidos os SOAP e REST.

Os serviços SOAP obedecem a um *schema* de informação XML que é trocado entre o servidor e o cliente. Este é composto pelo nó raiz *Envelope* que contém o nó facultativo *header* e o nó obrigatório *body*. O primeiro, quando presente, deve ser declarado imediatamente antes do elemento *body* e conter informações sobre o pedido, como por exemplo autenticação. O segundo elemento contém o pedido propriamente dito; indica o método a invocar e os parâmetros deste. Este elemento pode conter um único elemento *fault* que serve para indicar informações acerca de erros que podem ter ocorrido durante o pedido.
Tanto o pedido como a resposta de um serviço SOAP devem estar sob o *schema* desta

[6] http://php.net/

especificação [5].

Por outro lado, os serviços REST não obedecem a nenhum protocolo de encapsulamento nem de representação de dados. Estes serviços podem utilizar XML, JSON ou qualquer outra exposição de dados que seja conveniente. Apesar de toda esta flexibilidade, existem boas práticas para o desenvolvimento: utilizando o protocolo HTTP(S) são utilizados, normalmente, os tipos POST, GET, PUT e DELETE correspondendo cada um, respetivamente, a uma operação CRUD: *create* (inserir novos dados) , *read* (leitura de dados), *update* (atualização de dados) e *delete* (remoção de dados) [6].

A utilização de serviços SOAP teria várias vantagens uma vez que existem ferramentas para geração de serviços deste tipo e a existência de um descritor de serviço (WSDL) permite que os dados complexos sejam automaticamente mapeados para objetos. No entanto, estes serviços obedecem a um *schema* XML o que faz com que o volume dados transferidos seja maior do que se fossem somente transferidos os dados realmente necessários. Este é um ponto muito importante porque o objetivo dos *webservices* é permitir a troca de dados com dispositivos móveis, nomeadamente, *smartphones*. Estes terminais têm uma capacidade de processamento limitada pelo que é preferível simplificar os dados ao máximo para evitar carga no processador, mas também reduzir a quantidade de dados transferidos devido ao plano de dados limitados que os utilizadores possam ter subscrito. Posto isto, será preferível a utilização de serviços REST que permitem que os dados sejam adaptados à necessidade da comunicação e que sejam escritos de forma otimizada para evitar muito processamento.

3.2.3 Base de dados

A base de dados é algo importante na aplicação uma vez que é onde os dados serão guardados. É importante que o SGBD tenha um bom desempenho e tempo de resposta para não comprometer a performance do sistema e manter tudo em pleno funcionamento.

O MySQL[7] é um SGBD gratuito criado pela Sun (agora Oracle) e que se encontra massificado, tal como o PHP, no mercado do alojamento *Web*. Está construído de acordo com as normas ACID (*atomicity, consistency, isolation, durability*). A atomicidade garante que

[7]http://www.mysql.com/

uma transação é concluída toda com ou sem sucesso, ou seja, não permite que parte dela seja executada com sucesso e outra parte falhe. Em suma, se alguma operação falha, tudo o resto falha também. A consistência garante que as restrições presentes serão respeitadas, como por exemplo, chaves primárias ou estrangeiras. O isolamento garante que uma transação está isolada das restantes e que não influencia nenhuma outra. A durabilidade garante que quando uma transação é confirmada (*committed*) na base de dados assim permanecerá e os dados serão seguramente guardados. O cumprimento destas fazem do MySQL um SGBD de confiança e com potencial para sustentar os dados de todo o sistema [7].

3.2.4 Plataforma móvel

No mercado corrente, existem várias tecnologias que permitem criar aplicações para dispositivos móveis, sendo que algumas delas estão associadas ao próprio sistema operativo (por exemplo: iOS[8], Android[9] ou Windows Phone 7[10]) e outras que estão disponíveis em várias plataformas (por exemplo: Java Micro Edition[11]).

O Android é uma das plataformas que mais tem crescido. Vários fabricantes têm utilizado este sistema nos dispositivos móveis (não necessariamente telemóveis ou *tablets*). Esta larga utilização tanto em marcas como em modelos têm permitido que estes se encontrem a preços acessíveis. Também a disponibilização de um SDK para diferentes SO e a publicação de *software* gratuitamente no seu *market* tem atraído muitos programadores que não cobram nada pelos seus programas e os utilizadores são também atraídos por este facto.
No entanto, isto tem causado uma grande fragmentação pois os fabricantes nem sempre atualizam os dispositivos, fazendo com que estabilizem numa versão atrasada do sistema operativo.

O iOS é o SO exclusivo para os dispositivos móveis da Apple: iPod Touch, iPhone e iPad. Sendo uma plataforma exclusiva, esta está bem otimizada para os terminais onde é executada e a fragmentação é mínima (atualmente cada dispositivo recebe duas atualizações *major*). Esta é uma vantagem para os programadores que já conhecem em que hardware o *software* irá correr, mas o SDK existe apenas para MacOS e para testar os desenvolvimentos num

[8]http://www.apple.com/ios/

[9]http://www.android.com/

[10]http://www.microsoft.com/windowsphone/

[11]http://www.oracle.com/technetwork/java/javame/index.html

dispositivo físico é necessária uma licença paga da Apple. Isto poderá fazer com que algumas pessoas não usem esta plataforma como ponto de partida nos desenvolvimentos.

O WP7 é a aposta da Microsoft no mercado dos *smartphones* para o público geral. Os SO anteriores, Windows Mobile, tinham uma interface um tanto complexa e pouco *user-friendly*. Esta nova plataforma assemelha-se bastante ao Android uma vez que são os fabricantes já existentes no mercado que escolhem utilizar este SO nos telemóveis e a atualização dos mesmos é da responsabilidade da própria marca. À semelhança do iOS, o SDK existe apenas para o SO da própria marca, ou seja, Windows neste caso. Também é necessária uma licença paga para testar os desenvolvimentos no dispositivo e para submeter as aplicações para o público.

O JavaME é a aposta da Sun (agora Oracle) para o mundo dos dispositivos móveis. Esta edição do Java corre nos mais diversos terminais e não apenas num determinado SO. Na verdade, as plataformas mais recentes (como as que foram comparadas em cima) não têm suporte para esta tecnologia, sendo a sua utilização possível apenas através de emuladores. A razão para esta ter caído em desuso foi a sua estagnação, fraco desenvolvimento e devido ao facto de ser tão genérica e com o objetivo de correr em tantos terminais que não era possível tirar partido dos que possuíam melhor *hardware*. Além disto, estas aplicações são executas numa máquina virtual, KVM, o que muitas vezes prejudicava o desempenho da mesma.

Todas as plataformas móveis apresentadas têm os seus prós e contras. As plataformas iOS, Android e WP7 permitem tirar melhor partido do *hardware*, mas estão dependentes do SO. Já o JavaME corre em vários dispositivos, mais frequente nos legados, mas é mais difícil tirar partido das características do terminal.

A Tabela 2 apresenta uma comparação entre as plataformas apresentadas.

	Android	iOS	WP7	JavaME
Linguagem	Java	Objective C	C#.net / VB.net	Java
SO	Android	iOS	WP7	Symbian OS, Palm OS, S40, ...
Fragmentação	Alta	Baixa	Baixa	Alta
Custo de terminal	Acessível	Alto	Alto	Acessível
Desenvolvimento gratuito em terminal	Sim	Não	Não	Sim (para *debug* é necessário versão própria)
Massificação	Alta	Alta	Baixa	Alta

Tabela 2 - Comparação de plataformas móveis

O interesse para este sistema é ter uma aplicação que tire partido das funcionalidades e *hardware* do dispositivo para proporcionar uma agradável utilização ao cliente. A utilização de JavaME prevê uma má experiência nos SO de *smartphones* mais populares uma vez que não oferece a mesma experiência que um programa nativo. É importante que se possa testar a aplicação num dispositivo físico para ter uma ideia correta do seu desempenho e interação com os dedos. A fragmentação é um fator relevante porque é importante fazer chegar o *software* ao número máximo de terminais, evitando assim exclusões, mas também permitir que este não se concentre apenas em telemóveis topo de gama cujo o seu custo seja alto. Dadas estas necessidades, a plataforma Android é a mais completa e que cumpre um pouco de todos os requisitos: é vendido em terminais de vários preços, é possível testar os desenvolvimentos gratuitamente no dispositivo e é bastante popular. Apesar da sua alta fragmentação, as versões mais recentes correm *software* construído para versões anteriores e a aplicação a ser construída pode ser dirigida para a versão 2.1 que funcionará em 98.3% dos dispositivos

A escolha das tecnologias para o desenvolvimento do sistema é uma escolha importante. Estas podem determinar o futuro de toda a arquitetura, como o desempenho, escalabilidade e até segurança. A utilização do PHP para os módulos *Web* e *webservice* e MySQL para persistência de dados permite um menor custo na compra de alojamento, garantindo também performance e segurança e, no caso do SGBD, o respeito das normas ACID que é um fator

bastante positivo e importante. A plataforma móvel que vai mais ao encontro do necessário é o Android que tem uma vasta oferta no mercado, é popular e permite testar os desenvolvimentos diretamente no dispositivo.

3.3 Frameworks

Usando as tecnologias apresentadas no capítulo anterior, foram implementadas as aplicações e serviços para que o utilizador interaja com o sistema e para que a aplicação móvel comunique com o servidor. Neste capítulo são apresentadas as estratégias, *frameworks* e decisões na construção dos módulos do sistema.

3.3.1 Módulo *Web*

O módulo *web* fornece o portal para interação através de um navegador de internet e os serviços REST que fornecem dados para aplicações de terceiros. O portal *web* permite que os utilizadores comuns consultem as lojas e promoções existentes no sistema enquanto que utilizadores com permissões específicas podem fazer a gestão de informação variada. Os serviços REST permitem a troca de informação entre o cliente móvel e o servidor. Apesar do portal *web* e serviços REST fazerem parte do mesmo módulo, estes podem ser instalados em servidores diferentes (por exemplo, para balancear a carga) ou desativar apenas um deles (por exemplo, se for detetada alguma falha de segurança nos serviços REST, estes podem ser desligados e manter o portal *web* a funcionar em pleno).
A construção deste módulo foi feito com recurso às *frameworks* que são apresentadas nas próximas secções.

3.3.1.1 Symfony 2.0

A *framework* Symfony 2.0[12] é uma *framework* MVC, escrita em PHP, que permite construir aplicações *web* desacoplando as diferentes camadas (apresentação, lógica e dados persistentes). O funcionamento desta *framework* segue sempre o mesmo fluxo desde o pedido do cliente até à devolução da resposta. As páginas são definidas como *resource path*, ou seja, terá um aspeto de URL *user-friendly* ao invés dos típicos URL que definem o caminho para um ficheiro. Por exemplo, para aceder à página de gestão de utilizadores utiliza-se o link "http://www.exemplo.pt/users" ao invés de, por exemplo, "http://www.exemplo.pt/users.php". Para isto, a *framework* interpreta o pedido e reencaminha-o para o controlador correspondente para que o pedido seja processado corretamente. A Figura 3 exemplifica o modo como isto acontece:

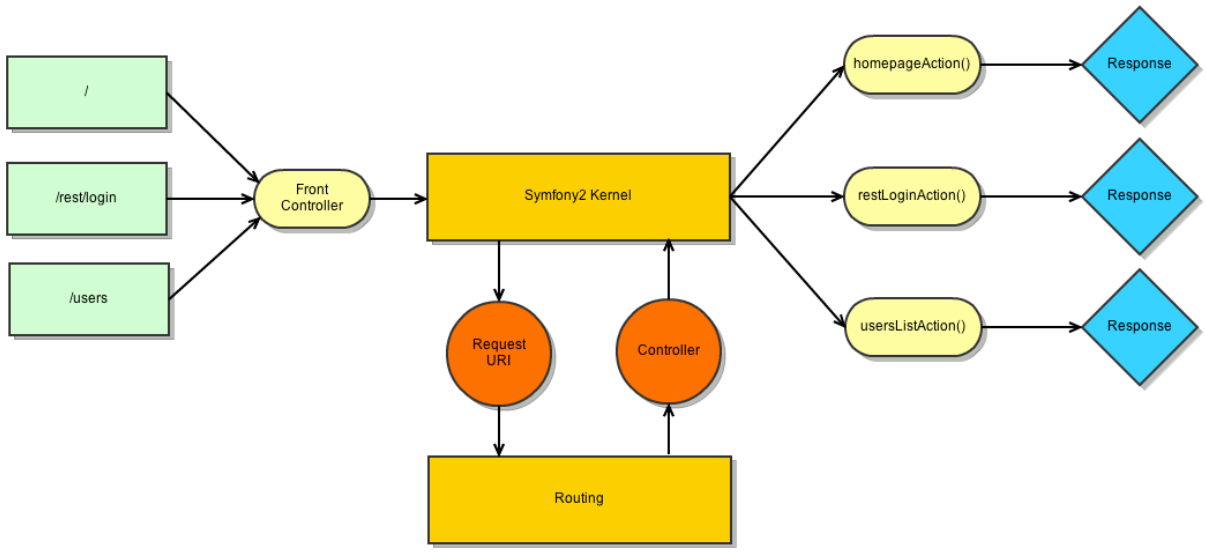

Figura 3 - Arquitetura interna da *framework* [13]

O pedido é feito ao *resource path* "/users" (por exemplo) e o *Front Controller* da *framework* recebe o pedido. Este *Front Controller* é um ficheiro PHP que trata o pedido para que seja corretamente processado pelo Symfony 2. Típicamente, este controlador é um ficheiro denominado de "app.php" (para ambientes de produção) ou "app_dev.php" (para ambientes de desenvolvimento). De forma a omitir o acesso a este ficheiro, pode-se utilizar o *mod_rewrite* do Apache para limpar e simplificar os URL.

Após o tratamento por parte do *Front Controller*, o núcleo da *framework* verifica se o

[12] http://www.symfony.com

[13] original in http://symfony.com/doc/current/book/http_fundamentals.html#the-symfony-application-flow

resource path acedido é válido e, caso sejam necessários parâmetros, verifica se estes são válidos. Esta validação é feita no módulo *Routing*, onde estão configurados todos os *resource path* disponíveis, os parâmetros necessários para cada um, os métodos HTTP permitidos, entre outros. Se o pedido for considerado válido por este módulo, é criada uma instância do controlador correspondente e invocado o método correto para que seja efetuada a lógica do lado do servidor e, por fim, devolvida uma resposta ao cliente.

3.3.1.2 Twig

Na secção anterior foi descrita a forma como a *framework* Symfony 2.0 atua perante pedidos dos clientes. No final de cada pedido deve ser devolvida uma resposta ao cliente para que este reconheça se o pedido foi efetuado com sucesso ou ocorreu algum erro. Esta resposta pode ser em vários formatos e, neste projeto, são utilizados dois tipos: HTML – no portal *web* – e JSON – para o cliente móvel.

A *framework* Twig[14] está integrada na Symfony 2.0 como motor de *templates*. Este motor permite agilizar o processo de construção das respostas através de *tags* especificas. Estas *tags* permiten, por exemplo, iterar uma lista de resultados e fazer decisões no fluxo da escrita. A utilização do *twig* é especialmente útil na construção de páginas HTML uma vez que este permite que exista um *template* principal onde é definido o cabeçalho, rodapé, menus e todos os outros elementos comuns a todas as páginas do *site*. Depois, para cada área do *site*, é definida uma página descendente do *template* principal. Esta funcionalidade é particularmente útil porque se for necessário fazer alguma alteração ao rodapé, por exemplo, esta tem que ser feita apenas no *template* principal e não em todas as páginas apresentadas no site.

3.3.1.3 Doctrine ORM

A *framework* Twig tem o propósito de apresentar a informação guardada na base de dados do sistema. O acesso a uma base de dados MySQL deve ser feito através de SQL mas, de forma a agilizar o processo, é possível utilizador *frameworks* ORM como é o caso da Doctrine[15]. Esta, tal como o Twig, encontra-se integrada na *framework* Symfony 2.0.

Através do mapeamento dos atributos das classes PHP, é possível gerar toda a estrutura da

[14] http://twig.sensiolabs.org/

[15] http://www.doctrine-project.org/

base de dados e camada de acesso a dados.

A utilização desta *framework* é feita sempre através do mesmo ponto de entrada: o *Entity Manager* (classe gestora de entidades). Através deste é possível obter dados da base de dados (tendo ou não condicionantes), persistir novos dados ou atualizar os já existentes. Esta *framework* permite, de forma *out-of-the-box*, obter dados encapsulando *queries* simples (por exemplo, sem *subqueries*, apenas dados da mesma tabela e condições apenas sobre a mesma). Caso seja necessário um nível mais elevando na obtenção de dados, as *queries* devem ser escritas em DQL (*Doctrine Query Language*) no repositório da entidade. Um repositório é uma classe associada a uma entidade mapeada no Doctrine que pode ter a implementação de métodos que usam a base de dados de forma mais avançada (por exemplo, através do DQL). A utilização dos repositórios não é obrigatória, mas é aconselhada para manter coerência na organização de código.

3.3.1.4 JQuery

O JQuery[16] é uma *framework* javascript que fornece várias funcionalidades úteis no desenvolvimento *web*. No contexto deste projeto, o JQuery é utilizado para manipulação do DOM (*Document Object Model*) e efetuar pedidos assíncronos ao servidor (AJAX – *Asynchronous Javascript And XML*) em várias páginas do portal *web*. Por exemplo, em todas as páginas existe um mecanismo de *pooling* que verifica se existem novas mensagens enviadas por algum consumidor.

3.3.1.4.1 JQuery UI

O JQuery UI[17] é uma *framework*, e ao mesmo tempo extensão para o JQuery, de apresentação que fornece vários componentes visuais que facilitam a interação dos utilizadores e não existem nativamente nos *browsers* (por exemplo, separadores e *popups*) ou que existem mas são visualmente mais agradáveis que os nativos (por exemplo, os botões). Esta *framework* é utilizada nas páginas *web* que contêm formulários para customizar os vários componentes HTML, como por exemplo, caixas de texto ou botões.

[16] http://jquery.com/

[17] http://jqueryui.com/

3.3.1.4.2 DataTables

O DataTables[18] é uma extensão para a *framework* JQuery, apresentada em 3.3.1.4, que oferece, por exemplo, as funcionalidades de tabela dinâmica, paginação, carregamento assincrono de dados, entre outros. Este *plugin* é utilizado em várias páginas do portal *web* para listagem de dados, como é o caso dos utilizadores, lojas, promoções, entre outros. Esta extensão é utilizada nas páginas *web* em que são apresentados dados em lista.

3.4 Conclusão

As tecnologias propostas foram escolhidas de forma a corresponder às necessidades de todo o sistema e que este seja suportado por uma base sólida.

A arquitetura de alto nível apresenta a forma como os clientes se ligam ao servidor e a utilização do SGBD. Se o cliente for um navegador *web*, então o acesso é feito a partir de um *website* construído com recurso a PHP; se o cliente for a aplicação móvel Android, então o acesso é feito através de *webservices* REST.

Através da utilização das *frameworks* apresentadas, será possível agilizar o processo de desenvolvimento uma vez que estas oferecem desenvolvimentos otimizados sobre as *frameworks* nativas de cada linguagem e facilitam a organização de código. Sem estas o processo de desenvolvimento seria possível mas seria mais demorado.

[18] http://datatables.net/

4 Implementação

4.1 Base de dados

Os dados apresentados em todo o sistema são persistidos numa base de dados MySQL para que seja mantido o estado das operações que os utilizadores executam, tanto no portal *web* como na aplicação móvel.

Para que seja possível os utilizadores se autenticarem na aplicação e criar novos, existe uma tabela onde estas são guardados e é incluída informação tal como o nome, nome de utilizador, palavra-passe cifrada e email. Estes utilizadores têm um conjunto de permissões que definem o seu tipo: administrador, gestor, lojista ou consumidor.
Na tabela de lojas está presente a informação básica sobre as lojas como o nome, morada, horário entre outros.
Cada uma das entidades anteriores, utilizador e loja, terá uma morada associada. Esta morada é construída com base no diretório geográfico que é composto por quatro tabelas: países, distritos, concelhos e localidades. Cada tabela destas está relacionada com a anterior para que cada país tenha os seus distritos que por sua vez tem os seus concelhos que por fim tem as suas localidades.
Uma vez que cada promoção apenas pode pertencer a uma loja, cada registo da tabela de promoções conta com identificador da loja a que pertence. Cada promoção tem um limite de resgates e é utilizada a entidade de resgates para guardar os códigos gerados durante este processo.

O DER (diagrama entidade-relacionamento) é apresentado no Anexo A. Este está dividido em várias partes por questões de espaço, pelo que serão apresentadas as tabelas agrupadas por funcionalidades.

4.2 Gestão de utilizadores

A gestão de utilizadores está apenas disponível aos que possuam as permissões necessários e, tipicamente, estas estão apenas associadas aos administradores da plataforma.
A gestão dos utilizadores inclui a adição, remoção, edição, ativação e desativação.

4.2.1 Listagem de utilizadores

Para que um utilizador possa listar todos os utilizadores existentes no sistema, é necessário que possua a permissão que permite o acesso a tal. Caso o utilizador possua também permissões para editar os utilizadores, cada entrada irá conter opções para edição, remoção, alteração de palavra-passe e ativação/desativação do utilizador.

A Figura 4 apresenta o ecrã de listagem de utilizadores.

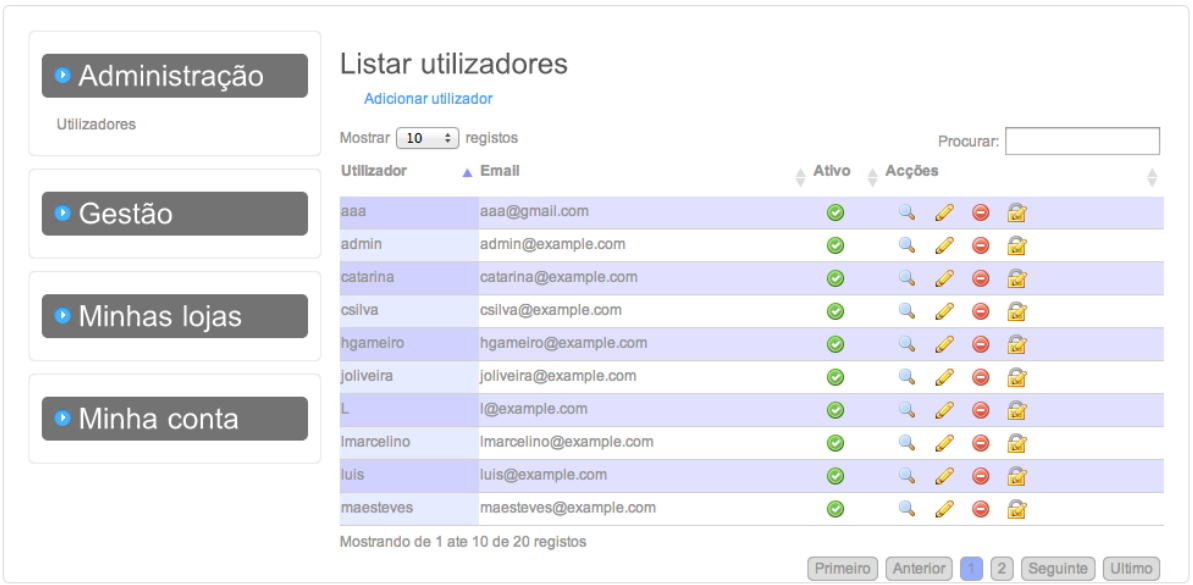

Figura 4 - Ecrã de listagem de utilizadores

4.2.2 Criar / editar utilizador

Um utilizador que tenha permissão para editar utilizadores pode também criar novos.
Ao ser aberta a página de criação de um utilizador, é necessário preencher os dados deste – incluindo as permissões a ele associadas – e ao gravá-los o utilizador passará a ter as permissões associadas e poderá autenticar-se no sistema.

A Figura 5 apresenta o ecrã de criação de utilizador

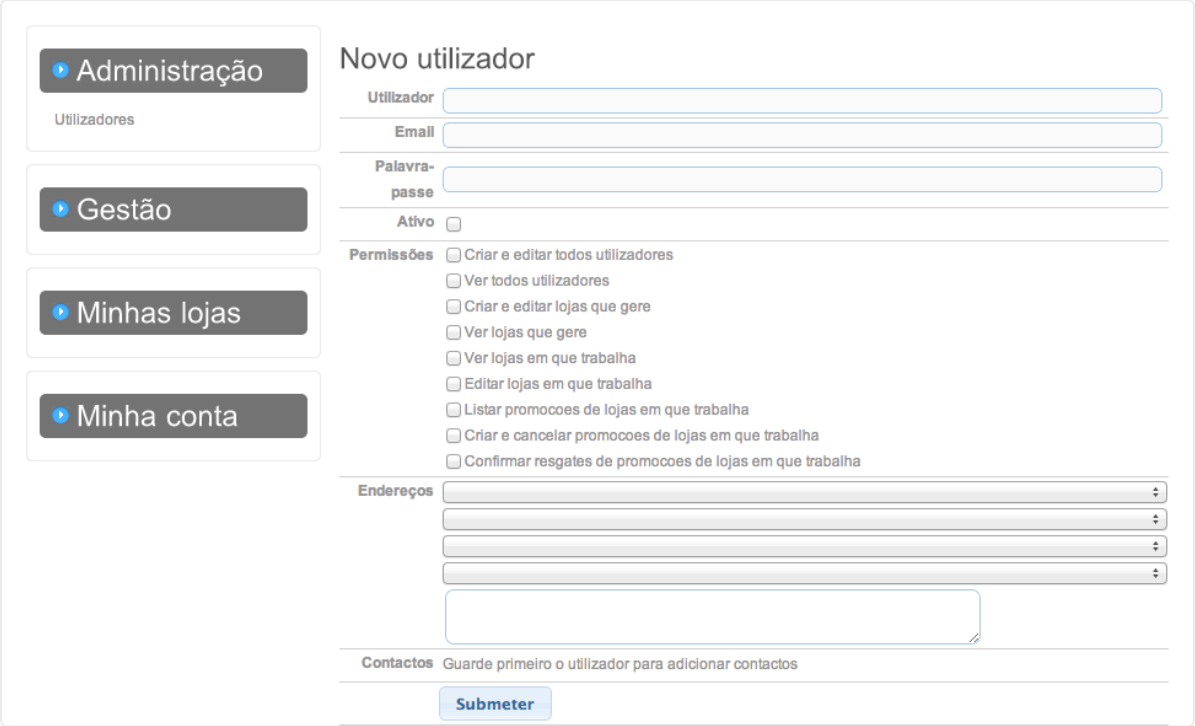

Figura 5 - Ecrã de criação de utilizador

4.2.3 Remoção de utilizadores

Para um utilizador estar apto à remoção de utilizadores, deverá ter a permissão que o permite editar utilizadores. Ao ser removido um, todos os dados ligados a ele serão também removidos. Como tal, quando for necessário remover um utilizador, o ideal é desativá-lo ao invés de remover.

4.3 Gestão das lojas

A gestão das lojas neste sistema é feita pelos utilizadores que possuam permissões para tal. Esta gestão inclui criação, edição e remoção das mesmas. Aquando de edição, é possível modificar todos os dados, adicionar e remover gestores e trabalhadores. Os gestores de loja são aqueles que podem modificar os detalhes da loja e, por essa razão, este tipo de utilizadores deveriam pertencer a uma associação de comerciantes, por exemplo, uma vez que a modificação destes dados deve ser monitorizada (por exemplo, a alteração da descrição da loja não pode conter publicidade ou difamação de outros estabelecimentos). Os trabalhadores são os utilizadores que podem gerir as promoções das lojas e responder a mensagens enviadas para a loja a que está associado.
As seguintes subsecções descrevem as operações inerentes a esta gestão.

4.3.1 Listagem de lojas

Um utilizador com permissão para listar as lojas pode consultar as lojas das quais é gestor. Todas as entradas têm uma opção de consulta de detalhes da loja e, caso o utilizador tenha permissões de edição das lojas, cada entrada terá a opção de edição e remoção.

A Figura 6 apresenta o ecrã de listagem de lojas

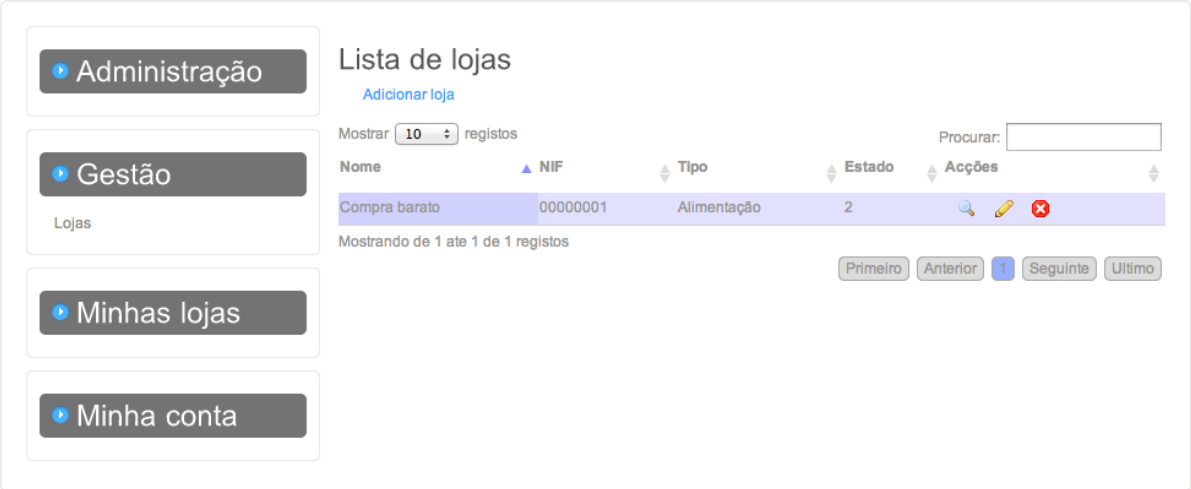

Figura 6 - Ecrã de listagem de lojas

4.3.2 Criação / edição de loja

Um utilizador que possua permissão para tal, pode criar e editar lojas. Ao adicionar uma loja, os campos desta apresentam-se vazios e após o preenchimento é possível guardar os valores ficando assim a loja adicionada no sistema. No caso de edição, o processo é o mesmo mas os campos já se encontram preenchidos com os valores atuais.

A Figura 7 apresenta o ecrã de edição de loja.

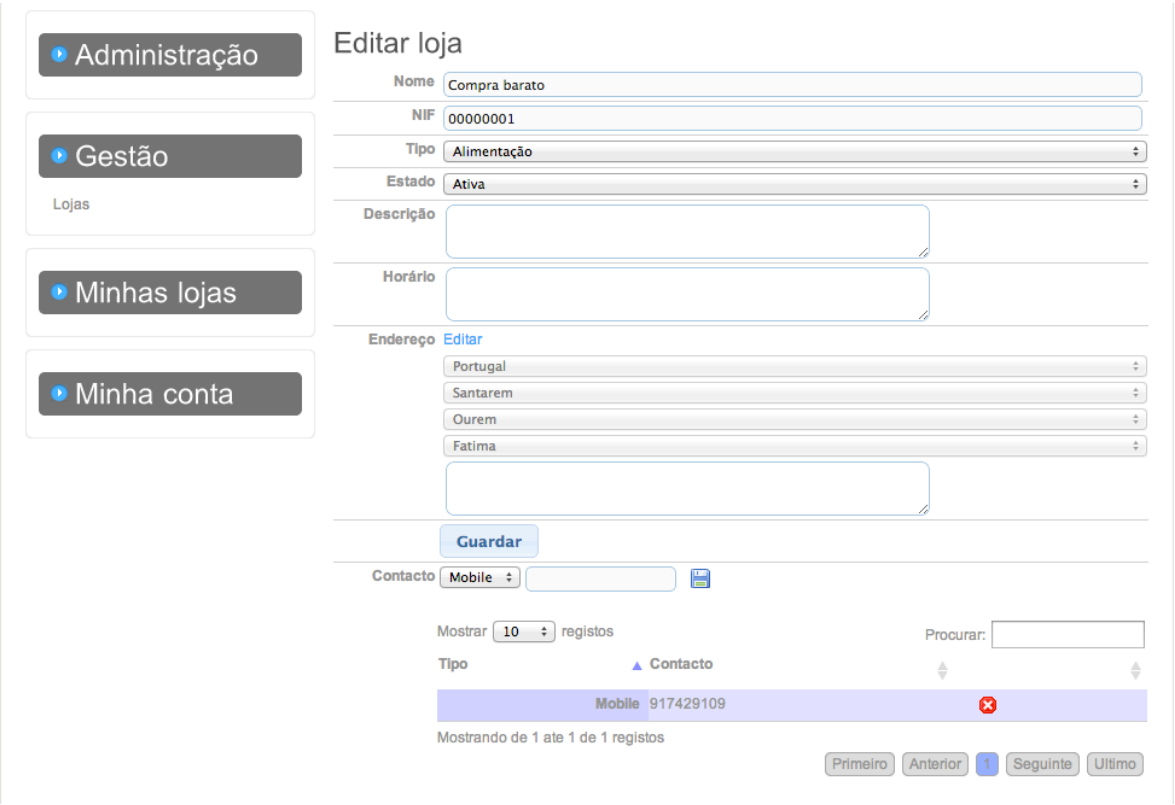

Figura 7 - Ecrã de edição de loja

4.3.3 Remoção de loja

Um utilizador que possa criar e editar lojas também está eligivel a remover lojas. Ao ser apagada uma loja, todos os dados são completamente removidos do sistema, ou seja, promoções, resgates, mensagens e outros dados ligados à loja removida são perdidos.

4.4 As minhas lojas

"As minhas lojas" é a área do site que permite aos lojistas fazerem a gestão das lojas às quais o utilizador está associado como lojista.

4.4.1 Listagem

Um utilizador que esteja adicionado como lojista a uma loja e tenha permissão para listagem das lojas, pode consultar as lojas a que está associado. A listagem a ser apresentada contém somente as lojas às quais o utilizador está adicionado como lojista.

A Figura 8 apresenta o ecrã de listagem das lojas das quais o utilizador está associado como lojista.

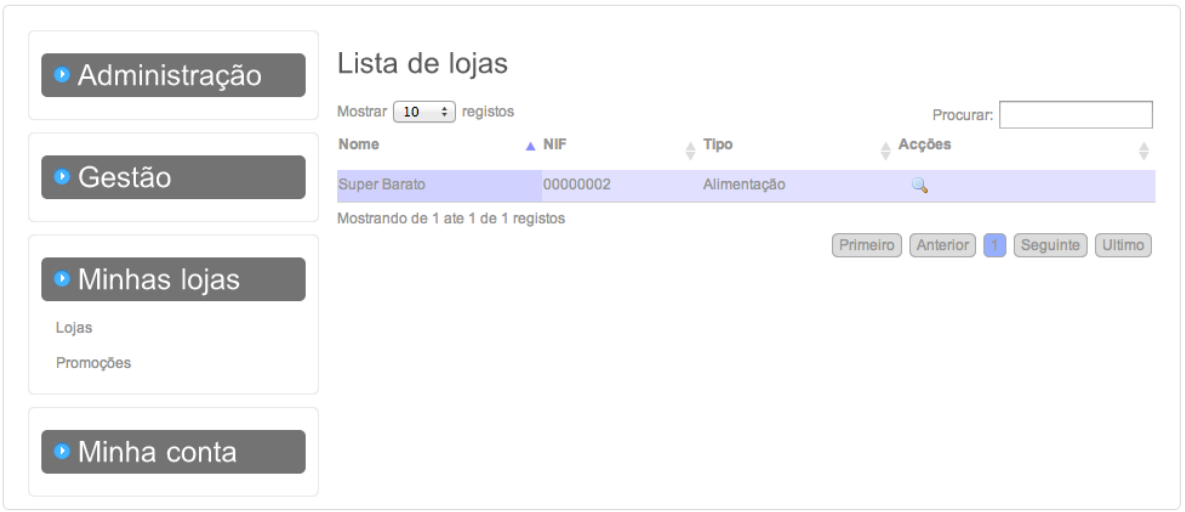

Figura 8 - Lista de lojas das quais o utilizador é lojista

4.4.2 Edição

Como trabalhador, o utilizador pode editar a loja somente se possuir a permissão para tal. Esta edição permite apenas modificar os contactos, morada e horário da loja.

A Figura 9 apresenta o ecrã simplficado para edição de uma loja.

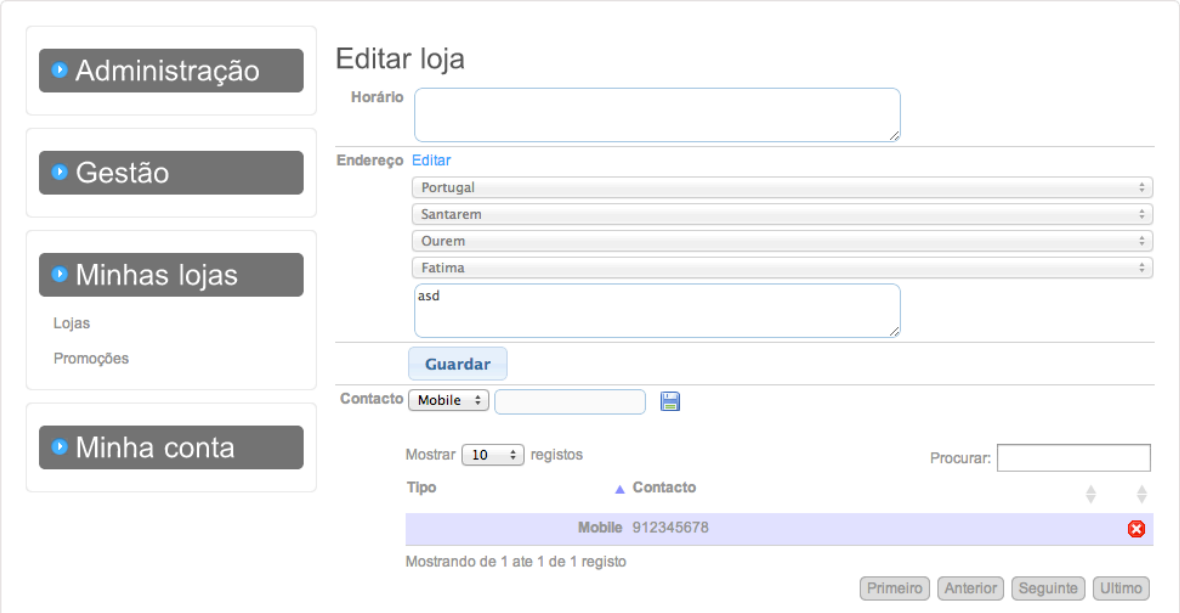

Figura 9 - Ecrã simplificado de edição de loja

4.5 As minhas promoções

"As minhas promoções" é a área do site que permite aos trabalhadores das lojas fazerem a gestão das promoções das próprias lojas.

4.5.1 Listagem de promoções

Um utilizador que possua a permissão para listagem de permissões pode listar as permissões das lojas em que está associado como trabalhador. Nesta listagem é apresentado o estado das redenções para cada promoção. Este estado inclui três partes: a primeira é o número total de redenções (estejam ou não confirmados), a segunda o número total de redenções confirmadas e por último o limite de redenções. Por exemplo, uma promoção cujo o limite são 10 redenções e já existem 6 das quais 3 estão confirmadas terá um estado semelhante a: "6/3/10".

A Figura 10 apresenta a listagem das promoções das lojas das quais o utilizador está associado como lojista.

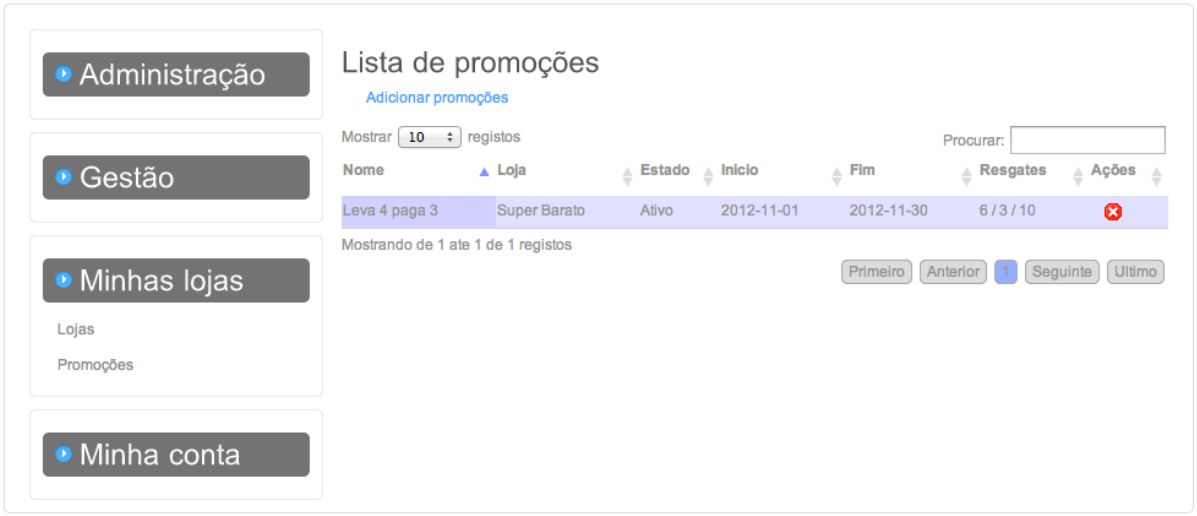

Figura 10 - Listagem de promoções das lojas do utilizador

4.5.2 Criação da promoção

Os trabalhadores com permissão para criar promoções, podem fazê-lo para as lojas das quais sao lojistas. Ao criar a promoção é necessário definir o número máximo de redenções confirmadas, data e hora de início e fim da promoção e em quais as lojas estará disponível. Para cada loja selecionada, será criada uma promoção distinta. Isto permite que, caso a promoção seja cancelada numa das lojas, as outras continuem com a promoção em vigor.

A Figura 11 apresenta o ecrão de criação de uma nova promoção.

Figura 11 - Ecrã de nova promoção

4.5.3 Cancelamento da promoção

No ecrã de listagem das promoções, e caso o utilizador esteja autorizado a cancelar promoções, também está disponível uma opção para cancelar a promoção. Aquando do cancelamento, será necessário dar uma razão para tal ter ocorrido. Após o cancelamento, os resgates não confirmados ainda podem ser confirmadas mas não serão aceites mais resgates por parte dos utilizadores.

A Figura 12 apresenta o *pop-up* para cancelamento de uma promoção.

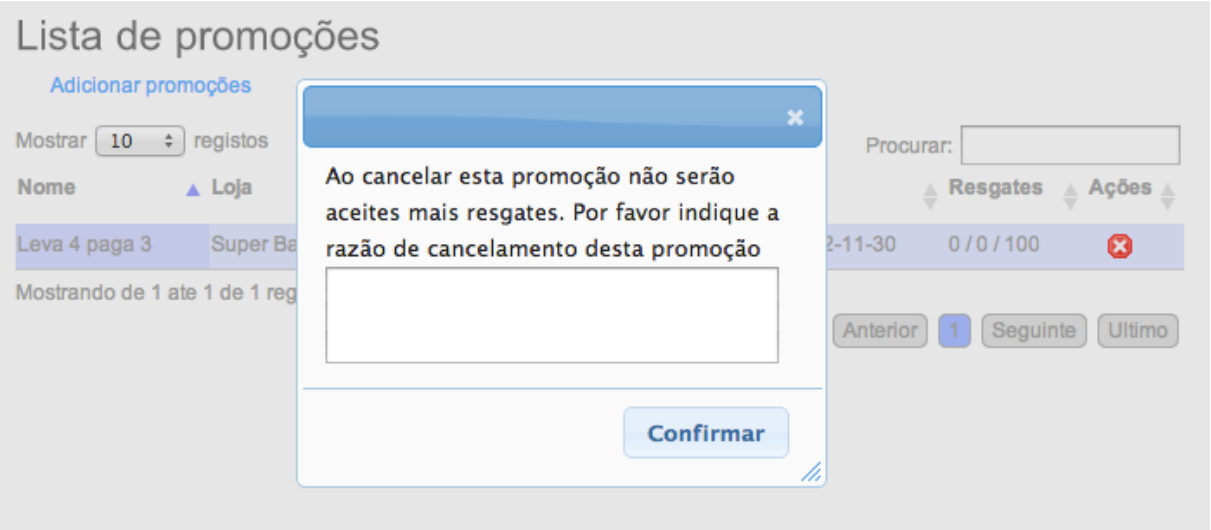

Figura 12 - *Pop-up* de cancelamento de promoção

4.5.4 Portal móvel

O portal móvel, fazendo parte do portal *web*, é onde o trabalhador da loja pode confirmar os resgates. Para aceder a este portal é necessário que o utilizador esteja autorizado a confirmar resgates de promoções. Aquando da confirmação, é necessário o utilizador selecionar se confirma o código ou se o rejeita. Se existir uma rejeição é necessário justifica-la.

A Figura 13 apresenta a página de entrada do portal móvel para confirmação de resgates

Figura 13 - Página de entrada do portal móvel

A Figura 14 apresenta a página de confirmação ou rejeição de um código de resgate válido.

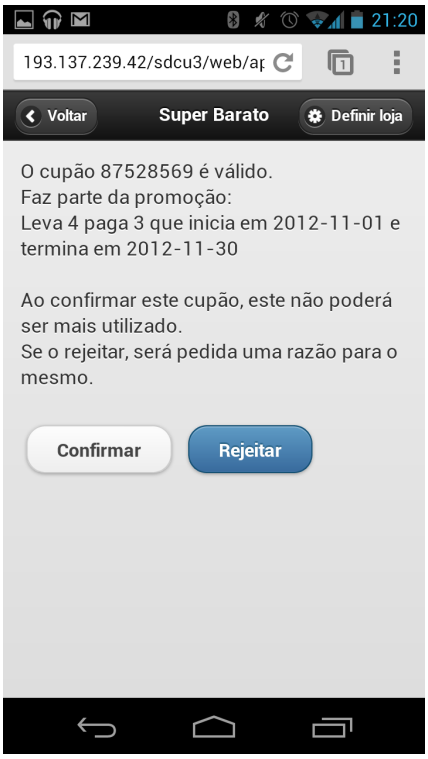

Figura 14 - Página de confirmação ou rejeição de um resgate

4.6 Serviços REST

Os serviços REST permitem que a aplicação móvel comunique com o sistema. Esta comunicação é importante uma vez que todos os dados estão guardados do lado do servidor e nenhum do lado do cliente. Assim, todos os dados consultados são os atuais e nunca estarão desatualizados.

Estes serviços recebem e fornecem a informação necessária para que o utilizador possa resgatar promoções, consultar os códigos

Todos os pedidos à API REST necessitam de dois elementos: uma API *key* e um *session token* (sendo este último não necessário na autenticação uma vez que esse processo fornece o *session token*). O URL base de acesso ao REST é:

http://<ip ou domínio>/api/<API key>/<session token>/<recurso>

4.6.1 Gestão de sessões

O *session token* permite que se mantenha um conceito de sessão em pedidos HTTP(S) independentes e sobre os quais não é mantido o estado. Através deste *token*, consegue-se saber quando foi o início da sessão do utilizador, quais as ações efetuadas durante a mesma e, se necessário, manter dados de sessão. Cada sessão tem um tempo de inatividade de 15 minutos. Ao ser efetuada a autenticação, se o utilizador não efetuar nenhuma operação em que seja feito um pedido ao servidor, então a sessão expira e é pedido ao utilizador que se autentique novamente. Cada vez que é feito um pedido ao servidor, o tempo da sessão é validado para os próximos 15 minutos. Estes *tokens* têm um comprimento de 128 caracteres numéricos para que a sua colisão seja pouco provável.

4.6.2 Funcionalidades

O serviço REST é o canal de comunicação entre a aplicação móvel e o servidor principal. Este serviço recebe e fornece dados do cliente de forma a que as ações do utilizador surtam efeito.

A obtenção do diretório geográfico (países, distritos, concelhos e distritos) permite que o utilizador procure as lojas por localização. Este também fornece os tipos de lojas para que seja possível o utilizador conhecê-las e facilitar-lhe a tarefa de encontrar os produtos que precisa.

O fornecimento da informação das lojas permite que o utilizador conheça os detalhes de cada uma incluindo as promoções que tem em vigor que são também fornecidas por este serviço.

Para facilitar o acesso às promoções, também é possível pesquisar promoções através das palavras-chave associadas às promoções.
Através deste serviço também é possível resgatar promoções, subscrever, e cancelar subscrição de, itens (do diretório geográfico ou tipo de loja). O envio e receção de mensagens entre os consumidores e lojistas é feito através deste serviço assim como a receção de alertas de novas mensagens e novas promoções pertencentes a algum item subscrito pelo utilizador.

No Anexo B encontra-se a documentação de todo o serviço REST.

5 Testes e Avaliação

Para que as aplicações, component *web* e móvel, sejam acessíveis aos utilizadores, é necessário efetuar testes para encontrar quais os pontos que se podem revelar confusos, enganadores ou pontos de entrada para incoerências em todo o sistema. De forma a testar estes pontos, foram feitos testes de vários tipos: testes funcionais e testes de usabilidade.

5.1 Testes de integração e funcionais

Para que seja possível construir um sistema resistente a falhas, é necessário que este seja testado de forma exaustiva. Para isto, existem vários tipos de utilizadores que podem efetuar diferentes tipos de testes e, de forma a que resulte valor destas ações, é necessário planear atempadamente quais os testes a serem efetuados. Neste caso estão contemplados três tipos de utilizadores para os testes: programador, *testers* independentes e clientes. O programador é quem desenvolveu o sistema e está responsável por implementar os requisitos. Os *testers* independentes são pessoas que estão à vontade com tecnologia, conhecem o sistema e a área de negócio mas não o desenvolveram. Os clientes são as pessoas para quem o sistema está a ser desenvolvido, não necessariamente o utilizador finalmente mas encaixam no mesmo perfil. Este não está necessariamente à vontade com tecnologia. Neste caso concreto, os clientes a testar esta aplicação serão lojistas e consumidores do dia a dia.
O planeamente destes testes atravessa várias etapas desde a definição do objetivo dos testes, definir os testes, escrever e testá-los e, por fim, executá-los e avaliar os resultados. Estes testes podem levar à identificação de falhas que devem ser corrigidas.

Numa primeira abordagem foram implementados, pelo programdor, testes de integração para garantir que a comunicação entre o servidor e o cliente é feita com sucesso e que os pedidos são interpretados corretamente em ambas as extremidades.
Os testes funcionais estão focados em testar a aplicação e verificar se esta está protegida contra incoerências. Este tipo de testes são efetuados pelos *testers* independentes e incluem tarefas como deixar campos obrigatórios em branco, escrever valores alfanuméricos em campos numéricos e, em exclusivo na aplicação móvel, desligar da internet durante uma comunicação com o servidor.
Os testes funcionais foram executados tanto pelos clientes como pelos *testers* independentes

para haverem mais casos de teste e reduzir ao máximo o número de falhas que podem passam despercebidas. Estes testes passam por utilizar a aplicação como se se tratasse do dia-a-dia. No caso dos lojistas todos os testes são feitos na plataforma *web* e o objetivo é gerir as promoções, confirmar o código que pertence ao resgate de uma promoção, adicionar e remover trabalhadores e responder a mensagens enviadas pelos clientes. Os consumidores utilizam a aplicação Android e o pretendido é navegar pelo diretório geográfico (países, distritos, concelhos e locais), tipos de lojas e lojas; resgatar promoções; pesquisar lojas e enviar mensagens. A administração é testada pela *web* e foi atribuída apenas a um *tester* porque qualquer alteração afeta todo o sistema e o seu objetivo é a gestão de utilizadores e lojas.

Após esta fase, são efetuados os testes de aceitação e *beta* cujo o objetivo e colocar o cliente final a utilizar a aplicação e testá-la ao mesmo tempo. Antes destes testes, é feita uma breve introdução de todo o sistema com o objetivo de apresentar melhor as funcionalidades e esclarecer algumas duvidas. Estes testes permitem receber *feedback* para alteração das funcionalidades, interface do utilizador e detetar falhas a serem corrigidas.

5.2 Testes de usabilidade

Apesar de haverem várias formas de efetuar testes de usabilidade e atualmente existem algumas metodologias bem conhecidas. No entanto, neste caso não foi seguida nenhuma metodologia específica. Os testes de usabilidade foram realizados por dez pessoas com idades compreendidas entre os 28 e 41 anos, todos eles ligados ao comércio de pequeno retalho. Quatro destas pessoas estão familiarizadas com a tecnologia e utilizam-na diariamente através de computadores e *smartphones* para variadas tarefas como acesso ao *homebanking*, elaboração da sua atividade laboral, acesso a redes sociais, entre outros. Dois dos *testers* utilizam a tecnologia apenas quando são obrigados a tal, como por exemplo, utilização de ATM e POS na loja. Os restantes quatro estão de alguma forma ligados à informática, uma vez que é a área de negócio dos mesmos; lidam com a tecnologia diariamente e é sobre esta que exercem a sua atividade laboral.

A cada *tester* foi fornecido um guião (Anexo 0) com as tarefas a realizar no âmbito dos testes e as credenciais a serem utilizadas para se autenticar no sistema. Em cada sessão de testes foram recolhidos vários dados que permitem tirar conclusões sobre o trabalho realizado e, no fim da sessão, foi realizado também uma pequena sessão de *brainstorming* com o âmbito de

obter *feedback* sobre a usabilidade e viabilidade que o *tester* vê no sistema.
Estes testes foram realizados nos mais variados ambientes, sendo a única garantia que o *tester* não seria interrompido. O facto de não ter sido preparado um ambiente isolado e controlado sem distrações permite que os testes se aproximem mais da realidade uma vez que é espectável que o sistema seja utilizado em ambientes mais ruidozos, como por exemplo, uma loja.

5.3 Discussão de Resultados

Após a fase de testes foram analisados os resultados que surgiram dos mesmos. Os parâmetros recolhidos, para cada tarefa, que permitiram extrair as conclusões foram:

- O tempo demorado, em segundos, até à conclusão da tarefa;
- O número de erros cometidos até à conclusão da tarefa;
- Indicação se o *tester* necessitou, ou não, de auxílio no decorrer da tarefa;
- Indicação se o *tester* conseguiu, ou não, concluir a tarefa com êxito, e
- As ideias partilhadas no fim da sessão de testes

A Tabela 3 apresenta as tarefas que constituem uma sessão de testes e a letra pela qual são identificadas.

A	Fazer login
Na loja "Compra barato":	
B	Edite a descrição para "Fruta fresca todos os dias"
C	Adicione o utilizador "rui" como gestor
D	Remova o utilizador "manuel" como gestor
E	Adicione o email "geral@comprabarato.pt" como contacto
F	Remova o número 917429109 dos contactos
G	Cancelar a promoção "Leve 3 pague 4" e indique "Nome errado" como razão de cancelamento.
Adicionar uma promoção	
H	Nome: "Leve 4 pague 3"
I	Descrição: "Na compra de 3 chocolates, oferta de um"
J	Início: 1 de Outubro de 2012
K	Fim: 30 de Novembro de 2012
L	Limite de resgates: 150
M	Lojas: Compra barato
N	Palavras-chave: chocolate, oferta
O	Definir a loja atual como "Compra barato"
P	Confirmar o resgate do código apresentado

Tabela 3 - Mapeamento das tarefas para letras

A primeira coluna indica a letra correspondente ao teste. Estas letras são indicadas nos gráficos seguintes onde são comparados os tempos médios e número de erros cometidos.

O Figura 15 apresenta o resumo dos tempos tomados para concluir cada tarefa.

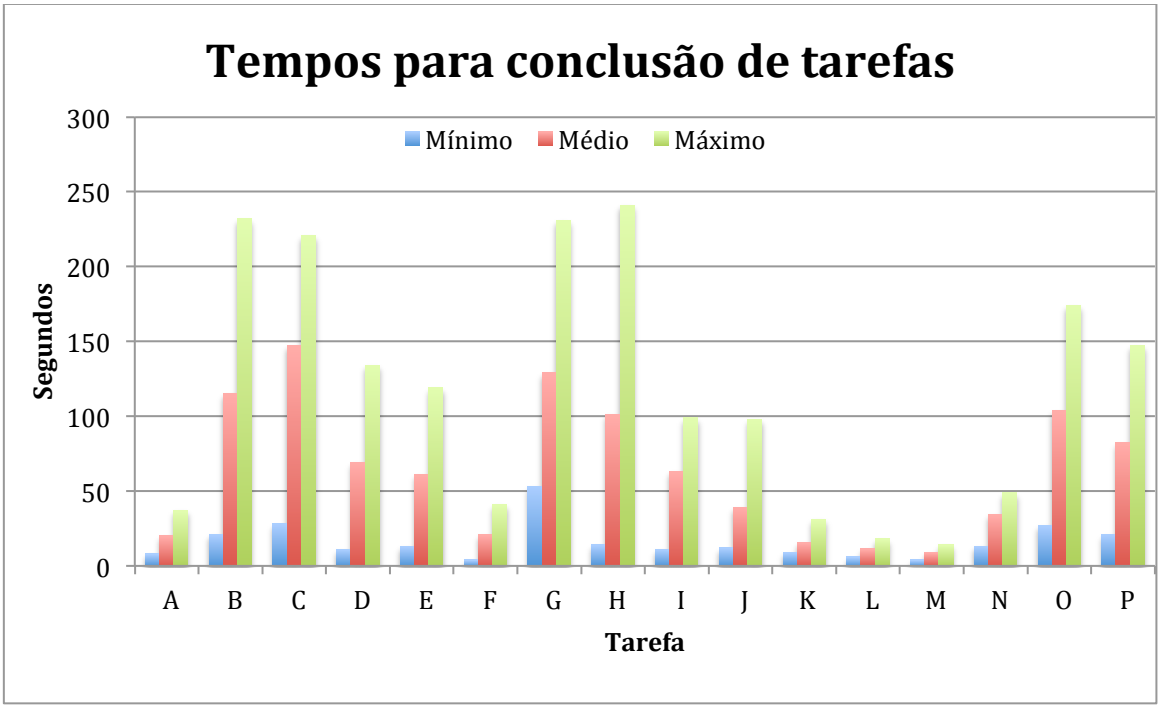

Figura 15 - Tempos de conclusão das tarefas dos testes

Este gráfico conta com três barras por cada tarefa. A barra azul apresenta o tempo mínimo que foi tomado por um utilizador para concluir a tarefa enquanto que a barra verde apresenta o tempo máximo. A barra vermelha mostra o tempo médio que os utilizadores levaram a concluir a tarefa. Através desta análise, é possível perceber quais as tarefas que demoraram mais tempo, na generalidade, a concluir. Estas são as que obrigavam os utilizadores a mudar de página e a interagir com ligações para abrir formulários como é o caso das tarefas B, G e H. À parte destas, as tarefas que demoraram mais a serem concluídas foram as que não são comuns em aplicações *web* mas que são específicas desta. Exemplo destas é a tarefa C, D, O e P. As tarefas que são comuns na maioria das aplicações, como autenticação ou preenchimento de formulários, foram rápidas e não causaram qualquer confusão aos *testers*.

Embora poucos, durante os testes os utilizadores cometeram alguns erros. A Figura 16 mostra um resumo dos erros por tarefa.

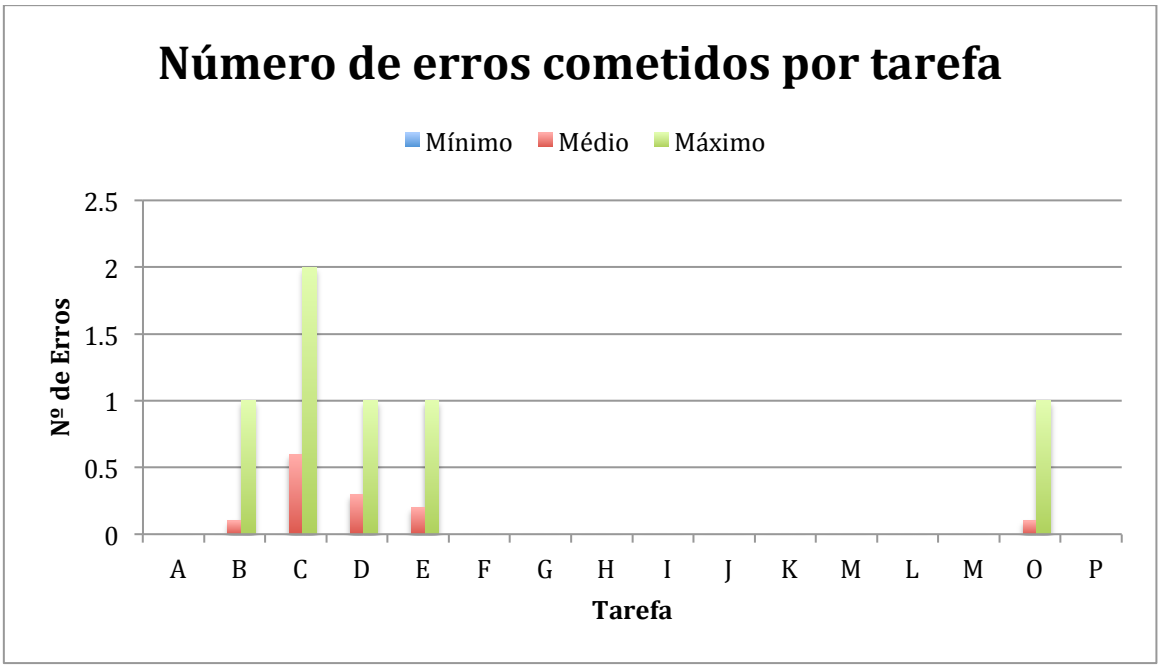

Figura 16 - Número de erros por tarefa na realização dos testes

À semelhança do gráfico na Figura 15, este conta com três barras por cada tarefa. A barra azul apresenta o número mínimo de erros cometidos pelos utilizadores durante o decorrer da tarefa, enquanto que a barra verde apresenta o número máximo de erros. A barra vermelha mostra o número médio de erros cometidos pelos utilizadores.

A ausência da barra azul mostra que em cada tarefa houve pelo menos um *tester* que não cometeu nenhum erro. Este é um fator bastante positivo uma vez que a ausência de erros motiva o utilizador a usar a plataforma uma vez que não tem de repetir a mesma tarefa para a fazer corretamente. Também é notável a ausência de qualquer erro em muitas das tarefas, o que indica que, nestes casos, nenhum utilizador cometeu erros. O facto de o valor médio de erros em cada tarefa ser abaixo de um, mostra que os erros cometidos não foram sistemáticos ou cometidos por todos; podem ter sido apenas erros causados por distração ou confusão mas que rapidamente foi ultrapassada.

A Figura 17 apresenta o tempo médio de cada tarefa comparando com a médio de erros cometido nessa mesma tarefa.

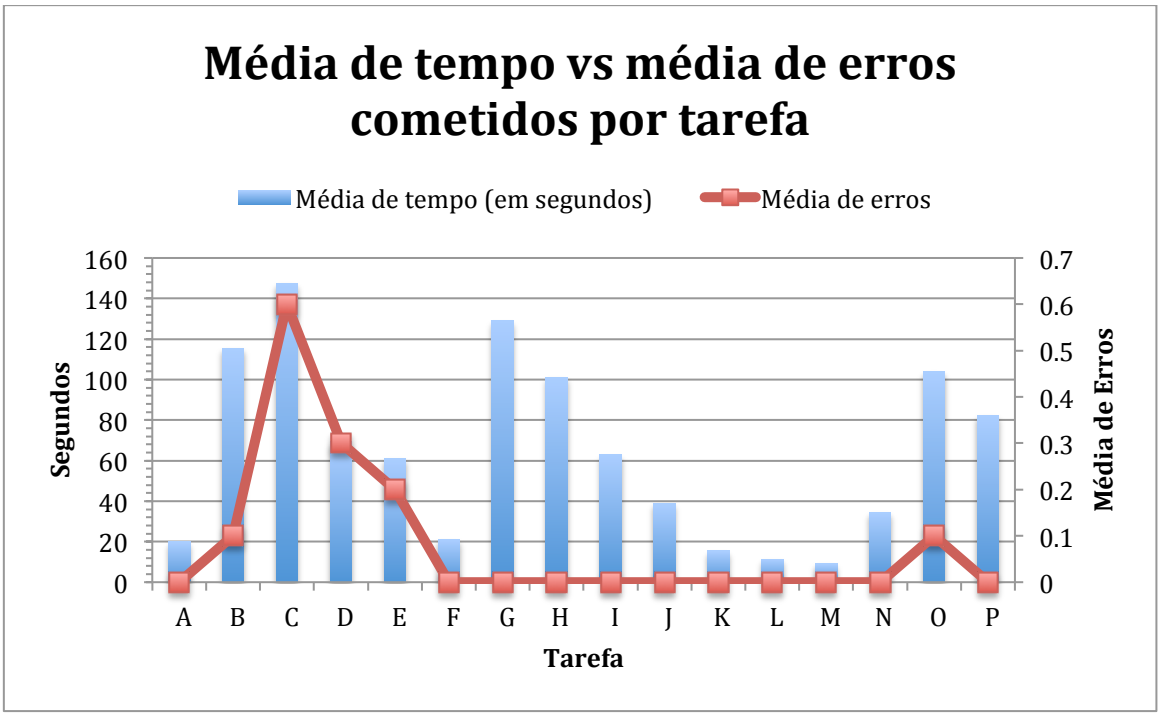

Figura 17 - Tempo médio para conclusão das tarefas VS número médio de erros

O gráfico de barras apresenta o tempo médio, em segundos, de cada tarefa e os linhas o número médio de erros para essa tarefa. As tarefas que implicam navegação pelo *site* são das que demoraram mais tempo e que, ao mesmo tempo, tiveram menos erros. Estas tarefas são as B, G e O. A razão destas demorarem mais tempo deve-se ao facto dos utilizadores necessitarem de mais tempo para analisar a página e decidir qual a hiperligação a seguir de forma a continuarem a tarefa. As tarefas que demoraram mais tempo e que ao mesmo tempo tiveram erros (C e D) são as que se implicam a interação com funcionalidades específicas desta plataforma e que se podem revelar confusas numa primeira abordagem. As restantes tarefas correspondem às tarefas comuns na maioria das aplicações *web*, como o *login* ou preenchimento de formulários.

A Figura 18 apresenta o tempo de cada *tester* e o número de erros que cada um cometeu durante a sessão.

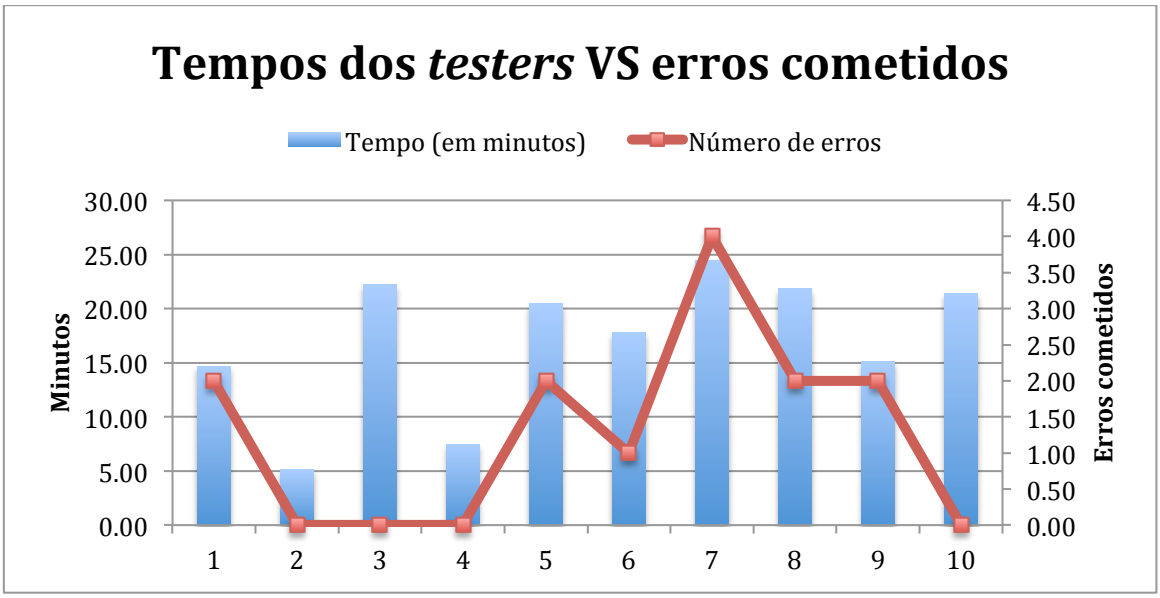

Figura 18 - Tempo de cada *tester* e erros cometidos

Os *testers* 2 e 4 trabalham como técnicos de informática e, por essa razão, é natural que sejam estes que tenham o melhor desempenho. Os *testers* 1 e 9 exercem atividade laboral também em loja de informática mas como gestores; esta vivência diária com a tecnologia permite que estejam mais à vontade com estas aplicações, mas mesmo assim cometendo erros. Os utilizadores 7 e 8 são pessoas que não estão familiarizadas com tecnologia e, por essa razão, são os que demoraram mais tempo e que cometeram mais erros em simultâneo. Os restantes quatro *testers* não tiveram uma oscilação muito significativa nos tempos nem no número de erros, o que mostra que o desempenho não foi acaso mas que foi coerente entre as diferentes sessões de testes.

Como foi referido anteriormente, no fim de cada sessão de testes foi pedido ao *tester* que desse o seu *feedback* sobre a plataforma. Todos os comentários recebidos foram positivos e de que o sistema será uma grande ajuda num mercado que está cada vez mais monopolizado pelas grandes superfícies. Os *testers* 1, 2, 4 e 9 identificaram o sistema como estando preparado para ser colocado em produção e que, ao longo do tempo, seriam feitas melhorias no mesmo. Os *testers* 3 e 10 mostraram muito interesse em toda a aplicação e que esperavam conseguir abranger uma maior área geográfica do que aquela que conseguem atualmente com prospetos em papel. Os *testers* 5 e 6 apelidaram a aplicação como um "potencial substituto do papel e fonte de poupança" uma vez que poderiam fazer a sua divulgação a partir da internet de forma gratuita ou com custos mais reduzidos do que os atuais prospetos, sinais

publicitários e publicidade em jornal. O *tester* 7 mostrou alguma reticência na viabilidade de substituir o papel pelo formato digital uma vez que o seu negócio tem um público alvo mais idoso, o que pode ser uma barreira à desistência do papel uma vez que esta faixa etária não estará tão à vontade com a tecnologia como seria desejado. Por último, os *testers* 8 e 10 ficaram bastante agradados com o conceito do *software* e da inovação que este poderá trazer ao mundo do pequeno retalho, sendo a única barreira a utilização da tecnologia por parte do público-alvo comum deste tipo de lojas; é um público de uma faixa etária mais avançada que não está habituada a lidar com tecnologia. Neste caso, a entrada deste sistema teria de ser feita de forma faseada, havendo divulgação pelo meio digital e tradicional em simultâneo e, lentamente, migrar para o meio digital. Durante esta fase, para incentivar a mudança, poderiam ser colocadas promoções exclusivas neste sistema de forma a aumentar a aderência ao mesmo.

6 Conclusão e Trabalho Futuro

Cada vez mais a população procura produtos baratos e tira proveito das promoções de forma a diminuir as despesas. Este facto leva a que as pessoas façam uma análise mais cuidada do que compram, onde compram e quando compram. Isto eleva as possibilidades das plataformas de divulgação de comércio terem mais sucesso uma vez que os consumidores começam procurar alternativas às mais populares.

A plataforma desenvolvida permite divulgar aos consumidores uma alternativa aos grandes grupos comerciais. Estas lojas de pequeno retalho, através de um atendimento personalizado e de preços igualmente moderados, conseguem fazer frente à crise que se faz sentir em todo o mundo. Face às plataformas apresentadas no estado da arte, esta permite uma maior aproximação do consumidor com o comerciante através do sistema de mensagens. O propósito de abrir esta plataforma de forma gratuita, ou com custos muitos reduzidos, é também um ponto a favor face às outras plataformas.

Com este trabalho, mostrou-se que existem retalhistas do comércio tradicional dispostos a utilizarem meios digitais para divulgação e comunicação e que isso poderá revelar-se uma mais valia pois permite abranger uma maior área geográfica e atrair novos clientes através de um atendimento mais personalizado. A única barreira diretamente identificada foi possível dificuldade em divulgar estes sistemas digitais pelo público mais idoso, que muitas vezes sente dificuldade em interagir com estes sistemas.

O contínuo desenvolvimento de qualquer *software* é o que lhe garante maturidade e uma notoriedade cada vez maior no mercado. Como tal, esta plataforma tem espaço a melhorias, tais como a inclusão de imagens nas promoções; a implementação de associação de produtos às promoções que não foi concretizada; a construção de uma aplicação móvel de referência para utilização por parte dos consumidores e a implementação de um portal *web* no âmbito do consumidor.

Bibliografia

[1] ***Article (Heijden2005)***
Heijden, H. V. D.; Kotsis, G. &Kronsteiner, R.
Mobile recommendationsystems for decisionmaking ‘ onthego ’
*Business,***2005**,137-143

[2] ***Article (Kowatsch2010)***
Kowatsch, T. &Maass, W.
In-storeconsumerbehavior: How mobile recommendationagentsinfluenceusageintentions, productpurchases, andstorepreferences
ComputersinHumanBehavior, **2010**, 26, 697-704

[3] ***Article (Westerman2007)***
Westerman, S.; Tuck, G.; Booth, S. &Khakzar, K.
Consumerdecisionsupportsystems: Internet versus in-storeapplication
ComputersinHumanBehavior, **2007**, 23, 2928-2944

[4] ***Article*** **(Cai2011)**
Cai, S. &Xu, Y.
DesigningNotJust for Pleasure: Effectsof Web Site AestheticsonConsumer Shopping Value
InternationalJournalofElectronic Commerce, **2011***, 15*, 159-188

[5] http://www.w3.org/TR/2007/REC-soap12-part0-20070427/#L1177

[6] ***Conference (Nunes2005)***
Nunes, S. & David, G.
XATA 2005 - XML: Aplicações e Tecnologias Associadas
Uma Arquitectura Web para Serviços Web, **2005**, 205-215

[7] http://www.databasejournal.com/features/mysql/article.php/3382171/Transactions-in-MySQL.htm

A. Base de Dados

A.1 Gestão de utilizadores

Para que seja possível o registo e autenticação dos utilizadores, é necessário que estes sejam guardados nas tabelas próprias para o efeito. A Figura 19 apresenta as tabelas relativas a esta parte do sistema.

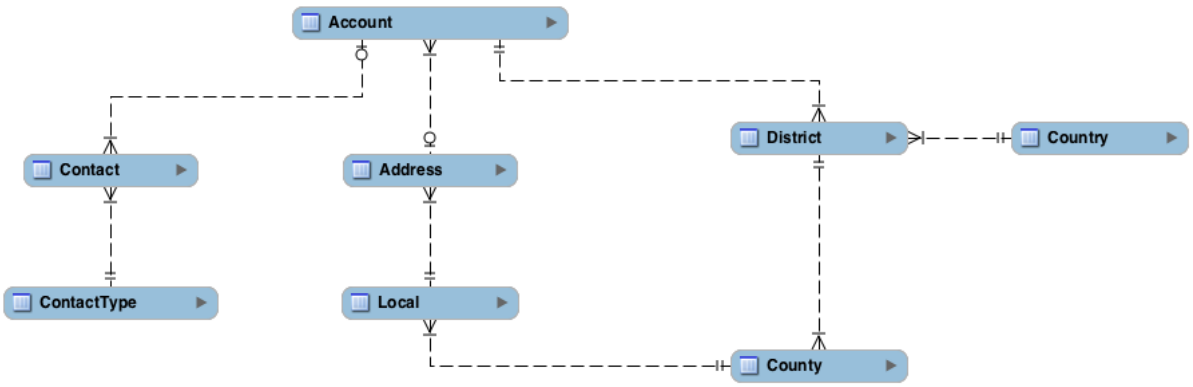

Figura 19 - Tabelas relativas à gestão de utilizadores

Tabela Country

Esta tabela, descrita na Tabela 4 , contém os dados relativos aos países existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
name	VARCHAR(255)	Nome do país.
insertedAt	DATETIME	Data e hora de criação do registo.
Flag	LONGTEXT	Imagem da bandeira do país armazenada em String Base64.

Tabela 4 - Detalhes da tabela Country

Tabela District

Esta tabela, descrita na Tabela 5, contém os dados relativos aos distritos existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
name	VARCHAR(255)	Nome do distrito.
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
idCountry	LONGTEXT	Chave estrageira para a tabela "*Country*". Indica a que país pertence este distrito.

Tabela 5 - Detalhes da tabela District

Tabela County

Esta tabela, descrita na Tabela 6, contém os dados relativos aos concelhos existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
name	VARCHAR(255)	Nome do concelho.
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
idDistrict	LONGTEXT	Chave estrageira para a tabela "*District*". Indica a que distrito pertence este concelho.

Tabela 6 - Detalhes da tabela County

Tabela Local

Esta tabela, descrita na Tabela 7, contém os dados relativos às localidades existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
name	VARCHAR(255)	Nome da localidade.
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
idCounty	LONGTEXT	Chave estrageira para a tabela "*County*". Indica a que distrito pertence este concelho.

Tabela 7 - Detalhes da tabela Local

Tabela Address

Esta tabela, descrita na Tabela 8, contém os dados relativos aos endereços. Cada utilizador (ou loja) pode ter associado um endereço. No caso dos utilizadores, não é necessário estes terem endereço associado.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
address	LONGTEXT	Descrição da morada (rua por exemplo)
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
idLocal	LONGTEXT	Chave estrageira para a tabela "*Local*". Indica a que localidade pertence esta morada.

Tabela 8 - Detalhes da tabela Address

Tabela ContactType

Esta tabela, descrita na Tabela 9, contém todos os tipos de contactos presentes no sistema. Estes tipos podem ser, por exemplo, contacto fixo e endereço de *email*.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Designation	LONGTEXT	Descrição do tipo de contacto
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
icon	LONGTEXT	Imagem para este tipo de contacto armazenada em String Base64.

Tabela 9 - Detalhes da tabela ContactType

Tabela Contact

Esta tabela, descrita na Tabela 10, contém os contactos presentes no sistema. Cada contacto pode estar associado a um utilizador ou a uma loja.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
value	LONGTEXT	Descrição do tipo de contacto
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada.
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica A que utilizador pertence este contacto.
idShop	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica a que loja pertence este contacto.
idType	INT(11)	Chave estrangeira para a tabela "*ContactType*". Indica qual o tipo deste contacto.

Tabela 10 - Detalhes da tabela Contact

Tabela Account

Esta tabela, descrita na Tabela 11, contém os dados dos utilizadores. À exceção do último campo, "idAddress", foram todos gerados pelo módulo de segurança da *framework*.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Username	VARCHAR(255)	Nome de utilizador
Username_canonical	VARCHAR(255)	Nome de utilizador
Email	VARCHAR(255)	Email do utilizador
Email_canonical	VARCHAR(255)	Email do utilizador
Enabled	TINYINT(1)	Indica se o utilizador está ativo
Salt	VARCHAR(255)	*Salt* para a palavra-passe
Password	VARCHAR(255)	Palavra-passe
Last_login	DATETIME	Data e hora da última vez que o utilizador se autenticou
Locked	TINYINT(1)	Indica se o utilizador está bloqueado
Expired	TINYINY(1)	Indica se as credenciais do utilizador expiraram
Expires_at	DATETIME	Data e hora de quando as credenciais do utilizador expiram
Confirmation_token	VARCHAR(255)	Código de confirmação para ativação do registo do utilizador
Password_requested_at	DATETIME	Data e hora de quando o utilizador pediu a redefinição de senha.
Roles	LONGTEXT	Lista de permissões, num formato legível pela *framework*, que o utilizador tem.

Credentials_expired	TINYINT(1)	Indica se as credenciais do utilizador expiraram.
Credentials_expire_At	DATETIME	Data e hora de quando expiram as credenciais do utilizador
idAddress	INT(11)	Chave estrangeira para a tabela "*Address*". Indica qual o endereço do utilizador.

Tabela 11 - Detalhes da tabela Account

A.2 Gestão de lojas

Para que possam ser inseridas promoções no sistema que os utilizadores possam resgatar, é necessário inserir lojas. A Figura 20 ilustra as tabelas relativas à gestão das lojas.

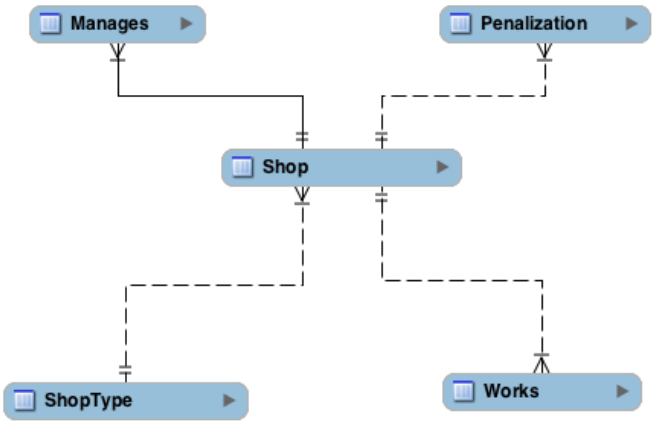

Figura 20 - Tabelas relativas à gestão das lojas

Tabela ShopType

Esta tabela, descrita na Tabela 12, contém os tipos de lojas existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Designation	LONGTEXT	Descrição do tipo de loja
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
Icon	LONGTEXT	Imagem para este tipo de loja armazenada em String Base64.

Tabela 12 - Detalhes da tabela ShopType

Tabela Penalization

Esta tabela, descrita na Tabela 13, contém as penalizações atribuídas às lojas.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Consequence	INT(11)	Consequência da penalização (enumeração definida no código)
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
reason	VARCHAR(500)	Descrição da razão da penalização.
idShop	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica a que loja esta penalização se refere.

Tabela 13 - Detalhes da tabela Penalization

Tabela Manages

Esta tabela, descrita na Tabela 14, contém a associação dos utilizadores com as lojas que podem gerir. Uma associação nesta tabela indica que o utilizador pode modificar todos os dados da loja.

Coluna	Tipo	Comentários
Account_id	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador gestor.
Shop_id	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica qual a loja gerida.

Tabela 14 - Detalhes da tabela Manages

Tabela Works

Esta tabela, descrita na Tabela 15, contém a associação dos utilizadores às lojas em que trabalha. Uma associação nesta tabela indica que o utilizador pode mandar mensagens em nome desta loja, gerir promoções e confirmar resgates (depende igualmente das permissões atribuídas a este).

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador trabalha na loja.
idShop	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica a que loja este trabalhador se refere.

Tabela 15 - Detalhes da tabela Works

Tabela Shop

Esta tabela, descrita na Tabela 16, contém todas as lojas existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
Status	INT(11)	Indica o estado da loja. (Estados definidos na implementação)
name	INT(11)	Nome da loja
Description	VARCHAR(600)	Descrição da loja
Picture	LONGTEXT	Imagem da loja armazenada em String Base64
Nif	VARCHAR(15)	Nº de Identificação Fiscal
Schedule	VARCHAR(500)	Horário
idAddress	INT(11)	Chave estrangeira para a tabela "*Address*". Indica qual a morada da loja.
idShopType	INT(11)	Chave estrangeira para a tabela "*ShopType*". Indica qual o tipo da loja.

Tabela 16 - Detalhes da tabela Shop

A.3 Gestão de promoções e resgates

O grande foco deste sistema é permitir os lojistas inserirem promoções das suas lojas e os utilizadores resgatarem-nas. O resgate é a emissão de um código exclusivo do utilizador para a promoção escolhida e que ficará inutilizável após ser utilizado.

A Figura 21 apresenta excerto do DER ustra a gestão das promoções e resgate das mesmas pelos utilizadores.

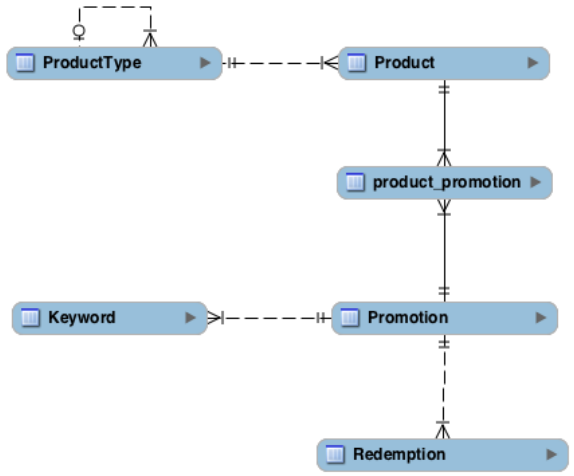

Figura 21 - Tabelas relativas à gestão de promoções e resgates

Tabela ProductType

Esta tabela, descrita na Tabela 17, contém todos os tipos de produto existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Parent	INT(11)	Chave estrangeira para a tabela "*ProductType*". Indica o pai do tipo de produto.
Designation	VARCHAR(255)	Descrição do tipo de produto
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
icon	LONGTEXT	Imagem para este tipo de produto armazenada em String Base64.

Tabela 17 - Detalhes da tabela ProductType

Tabela Product

Esta tabela, descrita na Tabela 18, contém todos os produtos existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
name	VARCHAR(255)	Nome do produto
Description	VARCHAR(500)	Descrição do produto
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
picture	LONGTEXT	Imagem para este produto armazenada em String Base64.
Barcode	VARCHAR(50)	Código de barras do produto
idProductType	INT(11)	Chave estrangeira para a tabela "*ProductType*". Indica o tipo de produto.
exclusiveTo	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica a que loja este produto é exclusivo. Se este campo não for preenchido, então não existe exclusividade do produto

Tabela 18 - Detalhes da tabela Product

Tabela Promotion

Esta tabela, descrita na Tabela 19, contém todas as promoções existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Name	INT(11)	Nome da promoção
Description	VARCHAR(255)	Descrição da promoção
insertedAt	DATETIME	Data e hora de criação do registo.
InsertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
redemptionLimit	INT(11)	Número máximo de resgates confirmados para esta promoção
startTime	DATETIME	Data e hora de início da promoção
endTime	DATETIME	Data e hora de fim da promoção
Active	TINYINT(1)	Indica se a promoção está ativa
closeReason	VARCHAR(600)	Razão de encerramento da promoção
idShop	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica a que loja pertence esta promoção

Tabela 19 - Detalhes da tabela Promotion

Tabela product_promotion

Esta tabela, descrita na Tabela 20, guarda a associação entre as promoções e produtos. Uma entrada nesta tabela indica que uma determinada promoção inclui um certo produto.

Coluna	Tipo	Comentários
Product_id	INT(11)	Chave estrangeira para a tabela "*Product*". Indica a que produto é feita esta associação.
Promotion_id	INT(11)	Chave estrangeira para a tabela "*Promotion*". Indica a que promoção é feita esta associação.

Tabela 20 - Detalhes da tabela product_promotion

Tabela Keyword

Esta tabela, descrita na Tabela 21, guarda as palavras-chave associadas a uma promoção. Estas palavras-chave são usadas na pesquisa de promoções.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Keyword	VARCHAR(255)	Palavra-chave
IdPromotion	INT(11)	Chave estrangeira para a tabela "*Promotion*". Indica a que promoção está associada esta promoção

Tabela 21 - Detalhes da tabela Keyword

Tabela Redemption

Esta tabela, descrita na Tabela 22, contém dos os resgates efetuados pelos utilizadores do sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Code	VARCHAR(8)	Código gerado para a redenção
redemptedAt	DATETIME	Data e hora de quando o utilizador fez o resgate
verifiedAt	DATETIME	Data e hora de quando o código foi validado
redemptionConfirmed	TINYINT(1)	Indica se a validação do código foi efetuada
Rejected	TINYINT(1)	Indica se o código foi rejeitado aquando da validação
rejectionReason	VARCHAR(600)	Razão de rejeição do código
insertedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador criou esta entrada
verifiedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador lojista validou o código deste resgate.
redemptedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que utilizador resgatou este código.
idPromotion	INT(11)	Chave estrangeira para a tabela "Promotion". Indica a que promoção se refere este resgate.

Tabela 22 - Detalhes da tabela Redemption

A.4 Funcionalidades dos serviços REST

Os serviços REST fornecem um conjunto de funcionalidades que estão apenas disponíveis a partir da aplicação móvel. Estes serviços são: serviço de mensagens, subscrições e sistema de alertas.

Com o sistema de mensagens é possível os utilizadores comunicarem com os lojistas através de um serviço de mensagens. Este serviço permite que os utilizadores iniciem uma conversa com vista a, por exemplo, tirar dúvidas, requerer um serviço especial ou confirmar *stock* de determinado produto. Neste caso, a comunicação dos utilizadores é feita através da aplicação móvel enquanto que os lojistas utilizam o portal *web* para envio de mensagens.

A subscrição permite que um utilizador subscreva promoções de determinado local geográfico (cuja granulidade deriva desde a localidade até ao país), tipo de loja ou loja em específico. O serviço de alertas completa as funcionalidades anteriores uma vez que é responsável por fornecer ao utilizador a lista de novas promoções ou novas mensagens existentes.

Para que seja possível um utilizador aceder à aplicação móvel, é necessário autenticar-se. Para manter o conceito de gestão, ao autenticar-se, o utilizador recebe um *token* que deverá ser enviado em cada pedido para que seja identificado qual o utilizador que está a efetuar o pedido. Para além deste *token*, deve igualmente ser enviado também uma chave de API. Esta chave tem o intuito de se poderem criar estatísticas sobre qual a aplicação que mais movimentação tem na plataforma e, caso necessário, bloquear os pedidos da mesma.

A Figura 22 mostra o excerto do DER mostra as tabelas referentes a esta funcionalidade.

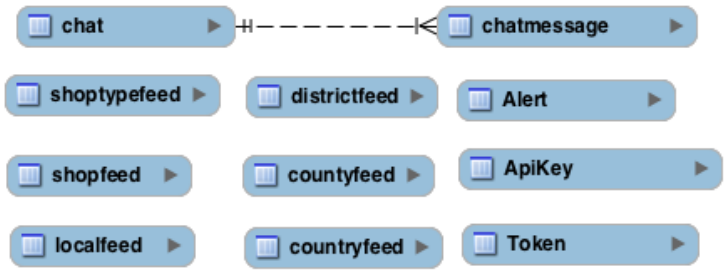

Figura 22 - Tabelas relativas ao serviço REST

Tabela Chat

Esta tabela, descrita na Tabela 23, inclui todas as conversas existentes no sistema. Apesar das conversas serem entre o utilizador e loja, é um dos lojistas que responde.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
ShopId	INT(11)	Chave estrangeira para a loja "*Shop*". Indica com que loja o utilizador está a comunicar.
insertedAt	DATETIME	Data e hora de início da conversa
startedBy	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que iniciou a conversa

Tabela 23 - Detalhes da tabela Chat

Tabela ChatMessage

Esta tabela, descrita na Tabela 24, contém as mensagens enviadas entre o utilizador e o lojista.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Message	VARCHAR(4000)	Conteúdo da mensagem enviada
insertedAt	DATETIME	Data e hora de envio da mensagem
chatId	INT(11)	Chave estrangeira para a tabela "*Chat*". Indica a que conversa pertence esta mensagem
Shopper	INT(11)	Chave estrangeira para a tabela "*Account*". Indica que lojista enviou a mensagem.
userRead	TINYINT(1)	Indica se o utilizador leu a mensagem
shopRead	TINYINT(1)	Indica se o lojista leu a mensagem
fromUser	TINYINT(1)	Indica se foi o utilizador ou lojista que enviou mensagem

Tabela 24 - Detalhes da tabela ChatMessage

Tabela ShopTypeFeed

Esta tabela, descrita na Tabela 25, contém as subscrições de promoções para os tipos de loja.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
ShopTypeId	INT(11)	Chave estrangeira para a tabela "*ShopType*". Indica que tipo de loja foi subscrito
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu este tipo de loja

Tabela 25 - Detalhes da tabela ShopTypeFeed

Tabela ShopFeed

Esta tabela, descrita na Tabela 26, contém as subscrições de promoções para as lojas.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
ShopId	INT(11)	Chave estrangeira para a tabela "*Shop*". Indica que loja foi subscrita
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu esta loja

Tabela 26 - Detalhes da tabela ShopFeed

Tabela CountryFeed

Esta tabela, descrita na Tabela 27, contém as subscrições de promoções para os países.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
CountryId	INT(11)	Chave estrangeira para a tabela "*Country*". Indica o país que foi subscrito
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu este país

Tabela 27 - Detalhes da tabela CountryFeed

Tabela DistrictFeed

Esta tabela, descrita na Tabela 28, contém as subscrições de promoções para os distritos.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
DistrictId	INT(11)	Chave estrangeira para a tabela "*District*". Indica o distrito que foi subscrito
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu este distrito

Tabela 28 - Detalhes da tabela DistrictFeed

Tabela CountyFeed

Esta tabela, descrita na Tabela 29, contém as subscrições de promoções para os concelhos.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
CountyId	INT(11)	Chave estrangeira para a tabela "*County*". Indica o concelho que foi subscrito
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu este concelho

Tabela 29 - Detalhes da tabela CountyFeed

Tabela LocalFeed

Esta tabela, descrita na Tabela 30, contém as subscrições de promoções para as localidades.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
LocalId	INT(11)	Chave estrangeira para a tabela "*Local*". Indica o concelho que foi subscrito
idAccount	INT(11)	Chave estrangeira para a tabela "*Account*". Indica qual o utilizador que subscreveu esta localidade

Tabela 30 - Detalhes da tabela LocalFeed

Tabela ApiKey

Esta tabela, descrita na Tabela 31, contém as chaves de API existentes no sistema.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
Apikey	VARCHAR(128)	Chave da API
Application	VARCHAR(255)	Nome da aplicação

Tabela 31 - Detalhes da tabela ApiKey

Tabela Token

Esta tabela, descrita na Tabela 32, contém os *tokens* gerados aquando da autenticação dos utilizadores pelos serviços REST.

Coluna	Tipo	Comentários
Id	INT(11)	Chave primária gerada sequencialmente.
token	VARCHAR(128)	*Token* de autenticação
sessionStart	DATETIME	Início da sessão
Validation	DATETIME	Data e hora de fim da sessão
active	TINYINT(1)	Indica se o *token* é válido
accountId	INT(11)	Chave estrangeira para a tabela "*Account*". Indica a que utilizador pertence este *token*.

Tabela 32 - Detalhes da tabela Token

B. Documentação serviço REST

B.1 Autenticação

A autenticação fornece o *session token* necessário para os restantes pedidos REST. A Tabela 33 apresenta a informação necessária para efetuar este pedido:

Caminho	**/api/<API *key*>/login**
Método HTTP	POST
Variáveis	username (string) password (string)

Tabela 33 - Descrição do pedido REST para autenticação

Como explicado anteriormente, o pedido de autenticação não necessita do *session token* uma vez que é devolvido, em caso de sucesso, por este pedido. O método HTTP a ser utilizado é o POST uma vez que são efetuadas modificações do lado do servidor (inserção de um novo *token* na base de dados). Por fim, para ser possível efetuar a autenticação, é necessário fornecer o nome de utilizador (*username*) e a palavra-passe (*password*) do utilizador.

B.2 Obter países

Uma vez que este sistema está preparado para funcionar com vários países, deve ser possível ao utilizador ver os países existentes e selecionar um dos países para ver os distritos existentes no país. A Tabela 34 apresenta a informação necessária para efetuar este pedido:

Caminho	**/api/<API *key*>/<*session token*>/countries**
Método HTTP	GET

Tabela 34 - Descrição do pedido REST para obtenção de países

A obtenção da lista de países é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.3 Obter distritos

Cada país pode ter vários distritos e estes podem ser obtidos através do pedido detalhado na Tabela 35:

Caminho	**/api/<API *key*>/<*session token*>/districts/<id país>**
Método HTTP	GET

Tabela 35 - Descrição do pedido REST para obtenção dos distritos de um país

A obtenção da lista de distritos de determinado país é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico do país.

B.4 Obter concelhos

Cada distrito pode ter vários concelhos e estes podem ser obtidos através do pedido detalhado na Tabela 36:

Caminho	**/api/<API *key*>/<*session token*>/counties/<id distrito>**
Método HTTP	GET

Tabela 36 - Descrição do pedido REST para obtenção dos concelhos de um distrito

A obtenção da lista de concelhos de determinado distrito é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico do distrito.

B.5 Obter localidades

Cada concelho pode ter várias localidades e estas podem ser obtidas através do pedido detalhado na Tabela 37:

Caminho	**/api/<API *key*>/<*session token*>/locals/<id concelho>**
Método HTTP	GET

Tabela 37 - Descrição do pedido REST para obtenção das localidades de um concelho

A obtenção da lista de localidades de determinado concelho é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico do distrito.

B.6 Obter lojas de localidade

Cada localidade pode ter várias lojas e estas podem ser obtidas através do pedido detalhado na Tabela 38:

Caminho	**/api/<API *key*>/<*session token*>/shops/local/<id local>**
Método HTTP	GET

Tabela 38 - Descrição do pedido REST para obtenção das lojas de uma localidade

A obtenção das lojas de determinada localidade é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico da localidade.

B.7 Obter promoções de loja

Cada loja pode ter várias promoções e estas podem ser obtidas através do pedido detalhado na Tabela 39:

Caminho	**/api/<API *key*>/<*session token*>/<id loja>/promotions**
Método HTTP	GET

Tabela 39 - Descrição do pedido REST para obtenção das promoções de uma loja

A obtenção da lista de lojas de determinada localidade é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico da loja.

B.8 Obter detalhes de promoção

Para se obter os detalhes de uma promoção, deve ser efetuado o pedido detalhado na Tabela 40:

Caminho	**/api/<API *key*>/<*session token*>/promotion/<id promoção>**
Método HTTP	GET

Tabela 40 - Descrição do pedido REST para obtenção dos detalhes de uma promoção

A obtenção dos detalhes de determinada promoção é efetuada através do pedido acima detalhado. Estes detalhes incluem o número de limite de resgates e quantos já foram efetuados, a descrição, entre outros. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro é o identificador numérico do distrito.

B.9 Obter tipos de loja

Cada loja está associada a um tipo e para se obter a lista destes deve ser executado o pedido detalhado na Tabela 41:

Caminho	**/api/<API *key*>/<*session token*>/shoptypes**
Método HTTP	GET

Tabela 41 - Descrição do pedido REST para obtenção dos tipos de loja

A obtenção da lista de tipos de loja é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.10 Obter lojas de tipo

Cada loja está associada a um tipo e para se obter a lista de lojas associadas a determinado tipo, deve ser executado o pedido detalhado na Tabela 42:

Caminho	**/api/<API *key*>/<*session token*>/shops/<id tipo>**
Método HTTP	GET

Tabela 42 - Descrição do pedido REST para obtenção de lojas de um tipo

A obtenção da lista das lojas de determinado tipo de loja é efetuada através do pedido acima detalhado. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET. O último parâmetro do pedido é o identificador numérico do tipo de loja.

B.11 Resgatar promoção

Para um utilizador resgatar uma promoção deve ser executado o pedido detalhado na Tabela 43:

Caminho	**/api/<API *key*>/<*session token*>/redeem/<id promoção>**
Método HTTP	POST

Tabela 43 - Descrição do pedido REST para resgatar uma promoção

Para se resgatar uma promoção deve ser executado o pedido detalhado acima. Ao ser pedido o resgate da promoção, e este efetuado com sucesso, o utilizador irá receber o código que poderá utilizar na loja para ter acesso à promoção. Como este pedido resulta em novos dados no servidor, o método HTTP utilizado é o POST.

B.12 Promoções resgatadas

É possível obter as promoções resgatadas por determinado utilizador. Para tal, deve ser executado o comando detalhado na Tabela 44:

Caminho	**/api/<API *key*>/<*session token*>/redemptions**
Método HTTP	GET

Tabela 44 - Descrição do pedido REST para obtenção das promoções resgatadas pelo utilizador

A obtenção das promoções resgatadas pode ser feita através do pedido em cima. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.13 Subscrever

É possível subscrever vários tipos de itens da plataforma tais como lojas, tipos de loja, localidades, concelhos, distritos e países. Para tal, deve ser executado o comando detalhado na Tabela 45:

Caminho	**/api/<API *key*>/<*session token*>/subscribe/<tipo>/<id item>**
Método HTTP	GET

Tabela 45 - Descrição do pedido REST para subscrição de um item

A subscrição de itens pode ser feita através do pedido detalhado em cima. O recurso "item" corresponde ao tipo de item a subscrever: loja (*shop*), tipo (*shop type*), localidade (*local*), concelho (*county*), distrito (*district*) ou país (*country*). O último recurso corresponde ao identificador numérico do item a subscrever.

B.14 Cancelar subscrição

Para cancelar a subscrição dos itens subscritos tal, deve ser executado o comando detalhado na Tabela 46:

Caminho	**/api/<API *key*>/<*session token*>/unsubscribe/<tipo>/<id item>**
Método HTTP	GET

Tabela 46 - Descrição do pedido REST para cancelamento de subscrição

O cancelamento de subscrição de itens pode ser feita através do pedido detalhado em cima. O recurso "item" corresponde ao tipo do item subscrito: loja (*shop*), tipo (*shop type*), localidade (*local*), concelho (*county*), distrito (*district*) ou país (*country*). O último recurso corresponde ao identificador numérico do item a ser cancelada a subscrição.

B.15 Pesquisar

É possível pesquisar promoções com base nas palavras-chave. Para isto, deve ser executado o comando detalhado na Tabela 47:

Caminho	**/api/<API *key*>/<*session token*>/search/<palavra-chave>**
Método HTTP	GET

Tabela 47 - Descrição de pedido REST para pesquia por palavras-chave

A pesquisa das promoções através de palavra-chave. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.16 Obter alertas

É possível obter os alertas em espera de serem lançados para determinado utilizador. Para obter estes alertas deve ser executado o comando detalhado na Tabela 48:

Caminho	**/api/<API *key*>/<*session token*>/alerts**
Método HTTP	GET

Tabela 48 - Descrição do pedido REST para obtenção de alertas lançados para o utilizador

O pedido detalhado em cima permite obter os alertas em espera de serem lançados a utilizador. Estes alertas podem ser de novas mensagens recebidas ou de novos itens subscritos. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.17 Obter conversas

Os utilizadores podem ter conversas com os lojistas e estas podem ser obtidas executando o comando detalhado na Tabela 49:

Caminho	**/api/<API *key*>/<*session token*>/chats**
Método HTTP	GET

Tabela 49 - Descrição do pedido REST para obtenção de conversas do utilizador

O pedido detalhado em cima permite obter as conversas associadas ao utilizador. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.18 Obter mensagens de conversa

Para se obterem as mensagens inerentes a uma conversa e deve-se ser executado o pedido detalhado na Tabela 50:

Caminho	**/api/<API *key*>/<*session token*>/chats/<id loja>**
Método HTTP	GET

Tabela 50 - Descrição do pedido REST para obtenção das mensagens de uma conversa

O pedido detalhado em cima permite obter as mensagens que foram enviadas para e recebidas pelos lojistas de determinada loja. O último parâmetro necessário é o identificador numérico da loja da qual se pretende obter as mensagens. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.19 Obter novas mensagens de conversa

Para se obterem as mensagens que foram recebidas de determinada após um determinado marco temporal deve ser executado o pedido detalhado na Tabela 51:

Caminho	**/api/<API *key*>/<*session token*>/chats/<id loja>/<*timestamp*>**
Método HTTP	GET

Tabela 51 - Descrição do pedido REST para obtenção de novas mensagens de uma conversa

O pedido detalhado em cima permite obter as mensagens que foram recebidas de determinada loja a partir de determinado marco temporal. O parâmetro "id loja" corresponde ao identificador numérico da loja sobre a qual pertencem as mensagens e o "*timestamp*" é um *timestamp* UNIX que corresponde ao marco temporal mínimo para recepção de mensagem. Uma vez que se trata apenas de obtenção dos dados, o método HTTP utilizado é o GET.

B.20 Enviar mensagem

Para se obterem as mensagens que foram recebidas de determinada após um determinado marco temporal deve ser executado o pedido detalhado na Tabela 52:

Caminho	**/api/<API *key*>/<*session token*>/chats/send**
Método HTTP	POST
Variáveis	shopId (integer) message (string)

Tabela 52 - Descrição do pedido REST para enviar mensagem

O pedido detalhado em cima permite que seja enviada uma mensagem para uma loja e esta poderá ser respondida por um lojista. Para que a mensagem seja enviada com sucesso, é necessário fornecer o identificador numérico da loja à qual é enviada a mensagem ("*shopId*") e a mensagem a ser enviada ("*message*"). Como esta operação resulta em adição de dados do lado do servidor, o método HTTP a ser utilizado é o POST.

C. Guião dos Testes

Guião

Obrigado por despender do seu tempo para testar o SDCU (Sistema de Divulgação de Comércio Urbano). Este sistema permite aos lojistas inserirem as promoções que praticam no seu estabelecimento tornando a sua divulgação digital permitindo, entre outras coisas, poupar papel e o capital que nele seria investido. Com esta divulgação, é possível os consumidores acederem à listagem de promoções e resgatarem um código que dará acesso à promoção que deverá ser confirmado pelo lojista no momento da aplicação da promoção.

O pretendido é avaliar a facilidade de uso da interface gráfica e não o utilizador. Mais uma vez, obrigado pela sua disponibilidade! A sua ajuda é imprescindível para o propósito destes testes!

Este guião apresenta de seguida o contexto em que os testes serão efetuados e uma lista das tarefas a serem executadas.

Contexto:

Você é um empregado da loja "Frescos e Companhia". Como empregado, você tem a possibilidade de fazer a alteração das informações sobre a loja, adicionar e cancelar promoções e confirmar códigos de promoções que os consumidores apresentem para resgatar. Esta loja permite que sejam feitas entregas ao domicílio na zona de Fátima entre as 16 horas e as 22 horas sempre que o valor total das compras atinja pelo menos os 15€. Durante os testes, irá receber mensagens de um cliente às quais deverá responder em função do contexto fornecido.

1. Fazer *login* na plataforma com o nome de utilizador e palavra-passe fornecidos. A aplicação deve ser acedida através de: http://goo.gl/M4csz

2. Na loja "Compra barato":
    a. Edite a descrição para "Fruta fresca todos os dias"
    b. Adicione o utilizador "rui" como gestor
    c. Remova o utilizador "manuel" como gestor
    d. Adicione o email "geral@comprabarato.pt" como contacto
    e. Remova o número 917429109 dos contactos

3. Cancelar a promoção "Leve 3 pague 4" e indique "Nome errado" como razão de cancelamento.

4. Adicionar uma promoção
    a. Nome: “Leve 4 pague 3”
    b. Descrição: “Na compra de 3 chocolates, oferta de um”
    c. Início: 1 de Outubro de 2012
    d. Fim: 30 de Novembro de 2012
    e. Limite de resgates: 150
    f. Lojas: Compra barato
    g. Palavras-chave: chocolate, oferta

Neste momento, um cliente apresentar-lhe-á um código proveniente da promoção que acabou de criar. Deve aceder a http://goo.gl/569IH para aceder à aplicação que permite a confirmação destes códigos.

5. Definir a loja atual como “Compra barato”

Confirmar o resgate do código apresentado