Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Oleksandr Novak

ESTÁGIO NA COMPTA

Relatório de Estágio

Orientado por:

Professor Doutor Pedro Correia, IPT – Instituto Politécnico de Tomar
Daniel Silva – Compta S.A.

Relatório de Estágio apresentado ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Produção de Conteúdos Digitais

RESUMO

Este relatório apresenta o trabalho realizado durante o estágio curricular desenvolvido durante o segundo ano letivo do Mestrado em Produção de Conteúdos Digitais. Este foi realizado na empresa Compta S.A, na área de desenvolvimento de aplicações web e mobile.

O trabalho realizado consistiu na melhoria da aplicação SouCidadão, que é uma aplicação mobile, cujo objetivo consiste em criar uma ligação entre o poder local e os cidadãos. Em particular, e enquadrado na equipa de desenvolvimento da aplicação, o trabalho consistiu na resolução de problemas existentes assim como a criação de novos serviços.

O segundo projeto descrito é a aplicação PaxVoice nas suas componentes *mobile* e *web*, aplicação que permite obter estudos de opinião sobre temas variados de interesse geral. O trabalho desenvolvido esteve enquadrado na componente web, onde foi implementado um sistema de autentificação por correio eletrónico.

O presente relatório descreve o funcionamento das aplicações, o trabalho desenvolvido assim como o processo de desenvolvimento das suas novas funcionalidades.

Palavras-chave: Drupal, Cordova, Ember.js, HTML, JavaScript

ABSTRACT

This report describes the work carried out during internship made in my second year of Master degree in Digital Content Production. I undertook this curricular internship at Compta S.A, with focus on development of web and mobile applications.

In particular, SouCidadão, which is a mobile application aiming to create communication links between local administration and citizens. In this application, the main contribution was to solve existing problems and also to create new services.

The second described work, is the application PaxVoice with web and mobile components, which provides opinion surveys on a variety of topics of general interest. The work was focused in development of web component, in particular, creating new email authentication system.

This report explains applications behavior and the process of development of their new functionalities.

Keywords: Drupal, Cordova, Ember.js, HTML, JavaScript

AGRADECIMENTOS

Em primeiro ligar queria agradecer à empresa Compta por me acolher durante estes nove meses de estágio e por me oferecerem esta oportunidade que se revelou uma ótima experiência.

Em específico queria agradecer ao coordenador de estágio Daniel Silva pela ajudou e apoiou durante este trabalho, e ao Nuno Miquelina, que me ajudou no desenvolvimento da aplicação de PaxVoice, pela disponibilidade que teve e pelos conhecimentos que me transmitiu.

Também queria agradecer ao meu orientador de estágio Pedro Correia por toda a ajuda.

Quero ainda agradecer a todos os colegas da Compta pelo acolhimento e partilha de conhecimentos.

Por último agradeço a todos os que me acompanharam ao longo deste percurso, em especial ao Dário pela companhia e à Márcia pelo apoio.

ÍNDICE

Índice de Figuras XV

Índice de Tabelas XVII

Lista de Abreviatura e Siglas XIX

1 Introdução 1

1.1 Enquadramento do Trabalho 1

1.2 A Empresa 1

1.3 Objetivos 1

1.4 Estrutura do Documento 2

2 Tecnologias 3

2.1 Drupal 3

2.2 Cordova 3

2.2.1 Instalação e Criação de Novo Projeto 4

2.2.2 Básicos 4

2.3 Linguagens de programação 4

2.4 Framework 5

2.5 Library 7

2.6 Bases de dados 8

2.7 Outros 8

3 Ambiente de Trabalho e Fases de Desenvolvimento 9

3.1 Software utilizado 9
3.1.1 Base de Dados 9
3.1.2 Edição de Imagem 9
3.1.3 Browsers 10
3.1.4 Virtual Private Network (VPN) 11
3.1.5 Git 11
3.1.6 Outros 11
3.2 Dispositivos de testes 12
3.3 Fases de Desenvolvimento 13
3.4 Migração para Servidor de Produção e Publicação de Aplicação 14
4 Aplicação SouCidadão 17
4.1 Introdução 17
4.2 Arquitetura da aplicação 18
4.2.1 Funcionamento de Drupal 19
4.2.2 Notificações 24
4.2.3 Serviço Recolha Monos e Verdes 29
4.2.4 Serviço Referendo Local 32
4.2.5 Outros Desenvolvimentos 36
4.2.6 Cache 36
4.2.7 Testes 37
5 PaxVoice 39

5.1 Introdução 39

5.2 Arquitetura da aplicação 39

5.3 Desenvolvimento 40

5.3.1 Resources 41

5.3.2 Sistema de Autentificação 42

5.3.3 APIs usados 42

5.3.4 Confirmações por email 44

5.3.5 Helpers 45

5.3.6 Registar 46

5.3.7 Login 48

5.3.8 Settings 49

5.3.9 Recuperação de Password 51

5.3.10 Ativação de conta e Reenvio de token 52

5.3.11 Múltiplas Línguas 52

6 Conclusões 53

7 Bibliografia 55

Índice de Figuras

Figura 1 - Diagrama de fases de Desenvolvimento de uma aplicação..... 13

Figura 2 - Processo de migração para servidor de produção e publicação nas plataformas 14

Figura 3 - Diagrama da arquitetura da aplicação de SouCidadão..... 19

Figura 4 - Um exemplo de vista de uma autarquia no Drupal. 20

Figura 5 - Um exemplo de Tipo de Conteúdo no Drupal, neste caso o conteúdo com nome de “Referendo” 21

Figura 6 - Exemplo de visualização no Drupal, na qual se representa uma visualização de “Page” com o respetivo “Título”, “Formato”, “Endereço” e outros campos editáveis pelo administrador. 22

Figura 7 - Exemplo de uma página Serviço no Drupal, na qual se representam na vista de autarquia, os serviços que são oferecidos, esta página tem um HTML estático com CSS adicionado..... 23

Figura 8 - Diagrama do processo de envio de notificação. 25

Figura 9 - Protótipo de layout das notificações "SouCidadão" em sistema Android. 28

Figura 10 - Resultado final das alterações na apresentação de notificações: 28

Figura 11 - Exemplo de uma lista de RMV de ponto de vista de uma autarquia. 30

Figura 12 - Seleção de recolha dos monos ou verdes no SouCidadão..... 31

Figura 13 - Segundo ecrã de serviço de recolha dos monos ou verdes no SouCidadão: 31

Figura 14 - Exemplo de uma lista de Referendos de ponto de vista de uma autarquia. 34

Figura 15 - Exemplo de detalhes de um Referendo de ponto de vista de uma autarquia. 34

Figura 16 - Ecrãs do serviço do Referendo Local 35

Figura 17 - Arquitetura do PaxVoice..... 40

Figura 18 - Email de confirmação de pedido de mudança de password 44

Figura 19 - Exemplo de auto completação do campo de Cidade 45

Figura 20 - Exemplo de um Helper com nome de “date-picker” 46

Figura 21 – Os ecrãs de registar no PaxVoice..... 47

Figura 22 - Template de ecrã do Login no PaxVoice..... 49

Figura 23 - Template dos Settings no PaxVoice 50

Figura 24 - A página de recuperação da password..... 52

Índice de Tabelas

Tabela 1 - Lista de informações de notificação antes de modificações. 25

Tabela 2 - Novas informações de notificação 26

Tabela 3 - Variáveis de “Tipo de Conteúdo” do RMV 29

Tabela 4 - Variáveis de “Tipo de Conteúdo” do Referendo 33

Tabela 5 - Variáveis de “Tipo de Conteúdo” do Resposta Referendo 33

Tabela 6 - Uma tabela do cache no SouCidadão 36

Tabela 7 - Tabela com os Resources criados no ficheiro routers.js 41

Tabela 8 - Os APIs usados no desenvolvimento de sistema de autentificação 43

Lista de Abreviatura e Siglas

API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
iOS	iPhone OS
JS	JavaScript
JSON	JavaScript Object Notation
Mac	Apple's Macintosh Computers
MVC	Model-View-Controller
MVVM	Model–view–viewmodel
RegExp	Regular Expression
REST	Representational State Transfer
RMV	Recolha Monos e Verdes
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell
SVG	Scalable Vector Graphics
UI	User Interface
URL	Uniform Resource Locator
VPN	Virtual Private Network
WinSCP	Windows Secure Copy
WP	Windows Phone
XML	eXtensible Markup Language

1 Introdução

1.1 Enquadramento do Trabalho

O presente trabalho foi realizado sob orientação de Daniel Silva, na empresa, e do Professor Doutor Pedro Correia, na modalidade de estágio, e faz parte integrante do ciclo de estudos conducentes ao grau de Mestre em Produção de Conteúdos Digitais, atribuído pela Escola Superior de Tecnologia de Tomar, do Instituto Politécnico de Tomar. O estágio decorreu entre novembro de 2016 e julho de 2017, na empresa COMPTA, S.A., no sector de *Business Solutions*, situado no edifício de *TagusValley* em Abrantes.

1.2 A Empresa

A Compta, foi fundada em 1972 e foi a primeira empresa tecnológica portuguesa. O principal foco da empresa é o desenvolvimento de tecnologias de Telecomunicações e Sistemas de Informação.

Atualmente, a Compta emprega mais de 200 trabalhadores. Acumula mais de 400 certificações individuais e detém o mais elevado nível de certificação nos seus principais parceiros estratégicos (Compta, s.d.a).

Business Solutions, o sector onde o estágio foi realizado, gestão de qualidade, financeira, vendas, *marketing*, produção, recursos humanos e logística e manutenção de aplicações (Compta, s.d.b).

1.3 Objetivos

O principal objetivo deste estágio foi o desenvolvimento e melhoria das aplicações SouCidadão e PaxVoice. No serviço SouCidadão, o trabalho foi direcionado para a introdução de novos serviços na aplicação. A plataforma SouCidadão, lançada em 2015, é uma aplicação *mobile*, que pretende criar uma ligação entre o poder local (Município/Freguesia) e os cidadãos, promovendo um contato mais próximo, fácil e ágil. A aplicação encontra-se disponível nas plataformas *mobile* mais usadas, *Android*, *iPhone OS* (*iOS*) e *Windows Phone (WP)*.

PaxVoice é uma plataforma social lançada em 2015, com objetivo de recolher e divulgar a opinião pública mundialmente sobre um dado tópico, em tempo real. Na aplicação PaxVoice, em resposta ao pedido dos utilizadores da aplicação, o trabalho focou-se na criação de um sistema de autenticação por email no website da aplicação.

O estágio focou-se ainda, na elaboração de testes de correto funcionamento das aplicações e na correção dos erros detetados.

1.4 Estrutura do Documento

Esta secção descreve a organização deste documento. O capítulo 1 apresenta o estágio e os seus objetivos, seguindo-se do capítulo 2 que descreve as tecnologias e *frameworks* utilizadas na fase de desenvolvimento do estágio. O capítulo 3 descreve o ambiente de trabalho, isto é, as ferramentas e as metodologias de trabalho usadas ao longo do estágio. O capítulo 4 apresenta a plataforma SouCidadão e o trabalho desenvolvido no âmbito do estágio. No capítulo 5 é utilizada a mesma abordagem para descrever o desenvolvimento do projeto PaxVoice. Finalmente o capítulo 6 comporta as conclusões do presente relatório.

2 Tecnologias

Este capítulo descreve as tecnologias e *frameworks* utilizadas no desenvolvimento dos projetos propostos pela empresa no decorrer do estágio.

2.1 Drupal

O *Drupal* é uma *framework* modular, *open-source*, escrita em *PHP*, com finalidade de criar e gerir aplicações web de modo simples e fácil.

Drupal oferece muitas vantagens, sendo *user-friendly*, *scalable* e *open source*, que significa que não existe custo monetário do uso deste *software*, e por este motivo, tem uma grande comunidade de utilizadores, que garantem que exista menor quantidade de *bugs*.

Esta *framework* também oferece módulos que podem expandir as suas funcionalidades, como por exemplo o módulo de *ColorBox*, que permite abrir um conteúdo do *Drupal* do tipo de Página numa janela (Drupal, 2016).

2.2 Cordova

Cordova, originalmente conhecido como *PhoneGap*, é uma *framework* para o desenvolvimento de aplicações multiplataforma, usando *HTML*, *CSS* e *JS*.

A *Cordova* cria um *WebView* que ocupa o ecrã inteiro, que por sua vez é executado em contentor nativo do *OS*. Isto significa que só o contentor nativo do *OS* é mudado conforme a plataforma usada, mas as páginas web corridas nela são as mesmas.

Esta *framework* fornece *plugins* diferentes que criam uma interface entre o *Cordova* e os componentes nativos, permitindo o uso de componentes de *hardware*, como a câmara, a *geolocation*, *splashscreen*, entre outros.

Apesar de *Cordova* ter muitas vantagens, também tem desvantagens, uma vez que não são aplicações nativas, ocupam mais espaço, são mais lentas e menos estáveis. Por vezes, ao solucionar problemas numa plataforma, surgem mais problemas noutras. No entanto os *smartphones* mais recentes já não são tão afetados, pois têm maiores recursos de processamento (Apache Cordova, s.d.).

2.2.1 Instalação e Criação de Novo Projeto

Antes de proceder à instalação de *Cordova*, é importante instalar o *Node.js* e o *SDK* do *Android*. Também é necessário ter os conhecimentos mínimos de funcionamento de linha de comando do *Windows*.

Para instalar o *Cordova* globalmente basta correr o comando:

```
$ npm install -g cordova
```

Antes de criar o novo projeto é necessário deslocar-se para o diretório onde o projeto se vai situar. Em seguida adicionamos o projeto correndo o comando:

```
$ cordova create [nome de aplicação]
```

2.2.2 Básicos

Após a criação do projeto, começamos por adicionar (também é possível atualizar ou remover caso a plataforma/*plugin* já existam) as plataformas e os *plugins* com os seguintes comandos:

```
$ cordova platform [add/remove/update] [nome da plataforma]
$ cordova plugin [add/remove/update] [nome de plugin]
```

2.3 Linguagens de programação

Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) e JavaScript (JS)

HTML, *CSS* e *JS* são as principais componentes utilizadas no desenvolvimento de *websites*, embora *HTML* e *CSS* possam não ser linguagens de programação uma vez que apenas servem para criar estrutura e o estilo das páginas web.

HTML é uma linguagem de marcação, usada para criar páginas de *World Wide Web*, que define a estrutura e *layout* do documento usando várias *tags* e atributos. Este *tags* e atributos são guardados num ficheiro ".html" que será lido pelo *browser* para mostrar as páginas web.

CSS é a componente responsável pela apresentação e aspeto da página web, tais como cores, fontes, *layout* e etc. Também permite configurar a página web para diferentes dispositivos, mudando o estilo da mesma conforme a resolução de ecrã.

JS é a componente de linguagem de programação no desenvolvimento de páginas web. O uso comum de *JS* é adicionar um comportamento *client-side* que permite dar às páginas elementos interativos ou animações que envolvem o utilizador.

PHP

PHP é uma linguagem *server-side*, criada principalmente para desenvolvimento web.

Os scripts *PHP* são executados no servidor e enviados para cliente como *HTML* simples, também pode ser incorporado nas páginas *HTML* normais.

2.4 Framework

Uma *framework* é um *software* que é criado para dar suporte no desenvolvimento de diferentes programas, no nosso caso, as aplicações *web*. A função de *framework* é ajudar a automatizar algumas tarefas.

Node.js

Node.js é uma plataforma que é usada para executar código de *JS* do lado do servidor. Esta *framework* foi criada para facilitar a construção das aplicações e torná-las mais escalável.

Angular.js

Angular.js é uma *framework open-source*, baseada em *Model–view–viewmodel* (*MVVM*) que serve para criar aplicações web dinâmicas, incluindo todos os mecanismos necessários para construir uma aplicação web completa no *front-end*. O *Angular.js*, permite criar as aplicações de página única, ou seja, uma aplicação que funciona dentro de *browser*, mas não necessita de voltar a carregar a página para mostrar novo conteúdo (Angular.js, s.d.).

Ionic v1

Ionic é uma *framework* para aplicações *mobile*, com maior foco na aparência e interação com *User Interface (UI)* da aplicação.

Esta *framework* foi construída com base em *Angular.js* e *Cordova*, permitindo criar aplicações *mobile* hibridas.

A parte de interação, gestos e animações requerem Angular.js, mas é possível usar a parte visual sem ter o mesmo (Ionic, 2017).

Ember.js

Ember.js é uma *framework* baseada construída em padrão de *MVVM*. Esta *framework* é muito semelhante a *Angular.js*, sendo

também usada para criar aplicações *front-end* e aplicações de página única (Ember, s.d.).

Bootstrap

Bootstrap é uma *framework* usada para criação de aplicações web, no entanto não oferece as mesmas funcionalidades como as *frameworks* mencionadas acima. *Bootstrap* é mais focado nos programadores que tem menos experiencia no desenvolvimento de *websites*, pois, essa *framework* não usa a arquitetura de *Model-View-Controller* (*MVC*), nem as funcionalidades como rotas, controladores e módulos.

2.5 Library

Uma *library* é um conjunto de recursos reusáveis, que tornam a programação mais fácil e eficiente. No decurso deste trabalho, a principal *library* utilizada foi *jQuery*.

jQuery

jQuery é uma biblioteca de *JavaScript*, cujo lema é "Escrever menos, fazer mais".

Esta biblioteca foi desenvolvida para simplificar scripts no *client-side*, e oferece as seguintes vantagens:

- Maior facilidade de navegação em documentos *HTML*;
- Criação de animações;
- Manipulação de eventos;
- Desenvolvimento de aplicações *AJAX* (jQuery, 2017).

2.6 Bases de dados

Base de dados é uma coleção de informação organizada para maior facilidade na gestão, acesso e atualização.

MySQL

MySQL é um sistema de gestão de base de dados baseado em *Structured Query Language (SQL)*, a vantagem de qual é o suporte de maior parte de sistemas operativos mais populares. É um dos sistemas de base de dados mais populares, normalmente usados mais no desenvolvimento web.

2.7 Outros

eXtensible Markup Language (XML) é uma linguagem de marcação usada para armazenamento ou transporte de dados. *XML* é muito semelhante a *HTML*, pois ambos podem usam as *tags* e atributos.

JavaScript Object Notation (JSON) é um formato de texto, normalmente usado para transmitir dados entre cliente e servidor. Este formato é fácil de ler e escrever, tornando-o um dos mais usados nos dias de hoje. A vantagem, comparando com *XML*, é ser possível funcionar com matrizes, ocupar menor espaço, oferecer mais facilidade de escrever e ler.

3 Ambiente de Trabalho e Fases de Desenvolvimento

Durante a realização do estágio, foi necessário o uso de diferentes ferramentas para realizar ou facilitar as tarefas propostas. Neste capítulo apresentam-se as ferramentas e dispositivos utilizados ao longo do estágio e a forma como foram utilizados. É também apresentado o fluxo de projeto usado no âmbito dos vários projetos realizados na empresa, assim como o trabalho desenvolvido no estágio para a integração das várias fases de desenvolvimento.

3.1 Software utilizado

3.1.1 Base de Dados

Ferramentas de bases de dados ajudam os programadores a visualizar mais facilmente as bases de dados e geri-las.

MySQL Workbench

É uma ferramenta ideal para desenho e gestão das bases de dados. Foi usada neste trabalho para visualizar e modificar base de dados de SouCidadão e PaxVoice, mas também na criação de tabelas que armazenam a cache do SouCidadão.

3.1.2 Edição de Imagem

Um editor de imagens permite-nos criar e modificar imagens de forma mais fácil e eficiente.

Paint.net

É um *software* de edição de imagem que substitui o *Paint* padrão do *Windows*. *Paint.net* tem uma interface mais intuitiva, possui mais funcionalidades e suporta diferentes *plugins* tornando-se um editor de imagens muito potente. Foi usado para editar todas as imagens no decorrer do estágio.

Inkscape

Inkscape é um *software* de edição de gráficos de vetores. Foi usado para edição e criação de ficheiros de *Scalable Vector Graphics (SVG)* nos projetos de SouCidadão e PaxVoice.

3.1.3 Browsers

Browser é um programa que permite navegar na web, processando as linguagens de programação tais como *HTML, JavaScript* e *CSS*, de forma a visualizar os conteúdos na forma de uma página web.

Durante o desenvolvimento dos projetos, foram usados diferentes *browsers* para testar o funcionamento das nossas aplicações web e aplicações *mobile*. Os testes de *mobile*, por serem feitos em emulação, não garantem que a aplicação funcione no *smartphone* físico, mas facilita o processo dos testes, tornando-o mais fácil do que correr a aplicação no telemóvel.

Os *browsers* usados para desenvolvimento móvel, em específico do SouCidadão e PaxVoice:

- *Chrome* – usado para emular android, e verificar o comportamento da aplicação em diferentes resoluções de ecrã;
- *Internet Explorer* – Usado para emular o *Windows Phone* 8 e 8.1;
- *Safari* – usado para emular o *iOS*, apesar dos *Apple's Macintosh Computers (Mac)* terem o emulador na aplicação de *xCode*, usá-la envolve copiar o projeto para o *Mac*, por esse motivo é mais fácil resolver os erros usando o *Bowser Safari* antes de avançar para o *Mac*.

No caso de aplicações web, nomeadamente PaxVoice, tentou-se garantir que todos os *browsers* populares funcionassem sem problemas, testando em *browsers* como *Chrome, Firefox, Opera, Safari, Microsoft Edge* e *Internet Explorer*.

3.1.4 Virtual Private Network (VPN)

VPN é uma rede de comunicação privada construída sobre uma rede de comunicações pública. O uso da *VPN* numa empresa tem benefícios como segurança, baixo custo, permite trabalhar remotamente, entre outros.

Check Point Endpoint Security VPN

Este programa foi usado para aceder à rede privada da Compta, possibilitando trabalhar nos servidores de desenvolvimento dos projetos como SouCidadão e PaxVoice, também permitindo aceder a outros recursos necessários no trabalho.

3.1.5 Git

Git é um sistema de controlo de versões de aplicações a serem desenvolvidas, este sistema permite a múltiplas pessoas trabalhar simultaneamente, partilhando o *source-code*.

GitKraken

GitKraken é um cliente de *git*, que dá suporte às tarefas como *push*, *pull*, *commit*, entre outras. Este cliente de *git* foi usado na aplicação de PaxVoice para obter a última versão e fazer o *push* das novas versões de *source-code*. O *source-code* do PaxVoice de momento encontra-se no website *BitBucket*.

Eclipse

Eclipse é um *Integrated Development Environment (IDE)*, mas no nosso caso foi usado para obter a última versão do *source-code* do SouCidadão, que de momento se encontra no *GitLab* da Compta, precisando da *VPN* para ter acesso.

3.1.6 Outros

Windows Secure Copy (WinSCP)

WinSCP é uma ferramenta que tem como função a transferência de ficheiros entre um computador local e um computador remoto. Este *software* foi usado na maior parte para editar os *web-services* e outros ficheiro do SouCidadão, mas, foi também usado no PaxVoice (WinSCP, 2017).

SoapUI

Este *software* é uma ferramenta usada para testar os *web-services*, em específico *Simple Object Access Protocol (SOAP)* e *Representational State Transfer (REST) API*. Foi usado no SouCidadão para fins de testes de *web-services* de uma forma fácil, pois não é necessário o uso de aplicação para o fazer (SoapUI, 2017).

Putty

Putty é um emulador de um terminal, que suporta protocolos como *Telnet*, *Rlogin* e *Secure Shell (SSH)*. Foi usado durante o desenvolvimento de SouCidadão e no servidor de qualidade do PaxVoice (PuTTY, 2017).

Xcode

Xcode é um *IDE* usado para gestão de projetos de *iOS* e *macOS*. Foi usado para correr/testar as aplicações SouCidadão e PaxVoice. esta aplicação só funciona no *Mac* (Apple, 2017).

3.2 Dispositivos de testes

Tal como foi dito anteriormente, parte dos objetivos deste trabalho de estágio incidiram em testes intensivos para garantir a qualidade das aplicações desenvolvidas.

Para efetuar estes testes utilizaram-se diferentes dispositivos móveis, sendo o *android* o mais testado, pois é o sistema mais usado para *smartphones*, seguido de *IOS*, que só necessita alguns ajustes para funcionar, e por último o *Windows Phone*, que representa apenas 1% do mercado de *smartphones*.

Os testes de funcionamento das aplicações nos diferentes *OS* foram efetuados com recurso a emuladores e aos seguintes equipamentos:

Android

- BQ Aquaris E5 (android 4.4)
- Nexus 4 (android 5.1)
- Nexus 5 (android 7.1)

- Asus Zenpad C (android 7.0)

IOS

- Iphone 4
- Iphone 5

Windows Phone

- Nokia Lumia 630

3.3 Fases de Desenvolvimento

Para garantir a maior qualidade no desenvolvimento das aplicações, antes do lançamento ao público, tem que passar por diferentes fases de desenvolvimento. A Figura 1, mostra um diagrama que ilustra as fases de desenvolvimento.

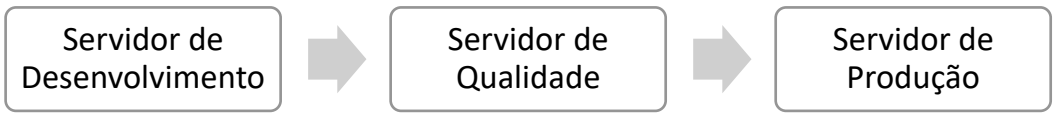


Figura 1 - Diagrama de fases de Desenvolvimento de uma aplicação.

Na Compta, para garantir a qualidade de processo, em primeiro lugar é necessário compreender a tarefa a realizar. Ao entender a função, procedemos a primeira fase de desenvolvimento, que é feita no Servidor de Desenvolvimento.

O Servidor de Desenvolvimento é um servidor dedicado para criar novos conteúdos de aplicações. Neste servidor não existem riscos de afetar os utilizadores, pois trata-se de uma rede completamente diferente e separada, onde as bases de dados não contêm os dados de utilizadores, mas sim, os dados redundantes, inseridos automaticamente ou manualmente. Enquanto as tarefas atribuídas não forem concluídas, o trabalho é realizado neste servidor, testando possíveis erros e reparando-os.

Após conclusão de desenvolvimentos e testes, são realizadas as migrações para o Servidor de Testes (SouCidadão neste momento não tem Servidor de Testes), que é uma réplica do Servidor de Produção, à exceção da base de dados que está preenchida com dados

falsos, só usados para teste. Neste servidor a tarefa principal é realizar testes intensivos para garantir que as aplicações tenham um comportamento correto e não existem erros ou problemas.

Na conclusão destes testes, garantindo que todos os problemas foram resolvidos, é feita a migração das modificações para o Servidor de Produção.

O Servidor de Produção é o servidor que é usado pelo consumidor final, ou seja, os utilizadores de aplicação. Nesta última fase, o desenvolvimento não acaba, pois é necessário continuar a corrigir os problemas que os utilizadores possam vir a encontrar, para garantir um bom funcionamento das aplicações no futuro.

3.4 Migração para Servidor de Produção e Publicação de Aplicação

Durante o desenvolvimento fiquei encarregado de migrar todas as mudanças do SouCidadão realizadas no Servidor de Desenvolvimento para o Servidor de Produção, este processo é mostrado na Figura 2.

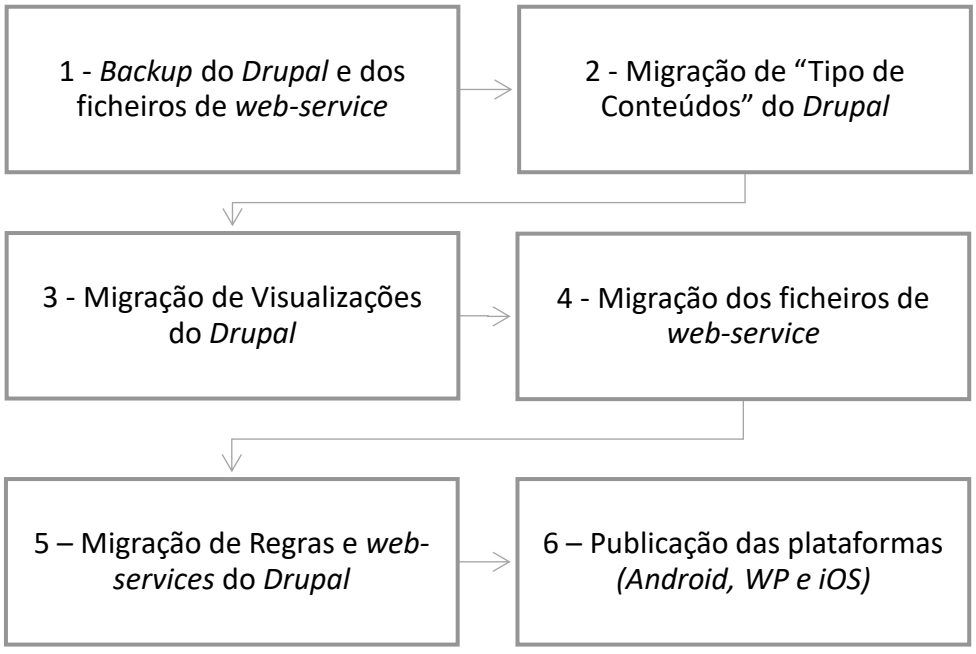


Figura 2 - Processo de migração para servidor de produção e publicação nas plataformas

Esta tarefa envolveu migrar todas as informações no *Drupal* relativas o "tipo de conteúdo", "visualizações", regras, web serviços e os ficheiros necessários para

funcionamento. Este processo necessita também fazer *backupts* antes de cada migração, garantindo que, se houver um erro, exista uma possibilidade de fazer *rollback*.

Após as migrações, o próximo passo consiste na publicação das últimas versões de cada plataforma *mobile (Android, iOS e Windows Phone)* e a realização de novos testes para verificar se todas as modificações tem funcionamento pretendido.

No *android*, o processo de publicação engloba gerar um ficheiro de formato ".apk" e assiná-lo com o seu certificado, depois de o ficheiro ".apk" estar assinado, pode-se proceder à publicação do mesmo. Neste processo acede-se na *Play Store* com conta da empresa, e escolhe-se a opção de criação de um novo lançamento. Dentro deste lançamento adiciona-se o ".apk" assinado, incrementa-se a versão de aplicação e adicionam-se as notas de lançamento (*release notes*) e finalmente faz-se o *update* para a nova versão.

No *Windows Phone* o processo é muito simples, bastando gerar o ficheiro de tipo ".xap", entrar na *Windows Phone Store* com a conta de empresa e escolher a opção de submeter uma atualização. Em seguida adiciona-se o ficheiro ".xap", incrementa-se a versão e submete-se o *update* (não é possível adicionar notas de lançamento no *Windows Phone Store*).

No *iOS*, as atualizações da versão é um processo mais complexo. Para fazer uma publicação, temos de usar um programa chamado *Xcode*, que só é suportado por sistema de *Mac OS*. Para começar é necessário aceder à *iTunes Store* e selecionar a nossa aplicação, em seguida escolher a opção de versões e plataformas e introduzir a nova versão incrementada. Procedendo, na aplicação de *Xcode* faz-se a *build* da aplicação, escolhe-se como dispositivo *Generic iOS device*, em menu de projeto, faz-se *Clean* e depois *Archive*. É escolhida a versão adicionada no *website* e submete-se a carregar no botão de "Upload to App Store". Volta-se à *iTunes Store*, seleciona-se a nova versão, e preenche-se os campos que faltam. Adicionam-se as notas de lançamento, e carrega-se no botão "Submit for Review".

No caso de existir um erro na submissão em alguma das plataformas, é nos enviado um email a informar a razão de erro, qual tem que ser corrigido para submeter a nova versão novamente.

4 Aplicação SouCidadão

Este capítulo apresenta a aplicação SouCidadão, o seu desenvolvimento e o trabalho realizado no âmbito do estágio.

4.1 Introdução

SouCidadão, lançado em 2015, é uma aplicação *mobile*, que pretende criar uma ligação entre o poder local (Município/Freguesia) e os cidadãos, promovendo um contato mais próximo, fácil e ágil. A aplicação encontra-se disponível nas plataformas *mobile* mais usadas, *Android*, *iPhone OS (iOS)* e *Windows Phone (WP)*.

Neste momento SouCidadão já esta a ser usado pela Junta de Freguesia de Porto Salvo, Junta de Freguesia de Mafra e Câmaras Municipais de Torres Novas e Abrantes.

De forma a estabelecer a ligação e promover a comunicação entre governo local e cidadãos, a aplicação pode disponibilizar varias funcionalidades (cada freguesia/município pode disponibilizar serviços diferentes), tais como:

- **Referendo Local** – O serviço possibilita, aos cidadãos, votarem nas ideias ou questões criadas pelo governo local, permitindo às autarquias saber a opinião dos seus cidadãos;
- **Recolha de Monos e Verdes** – Este serviço permite aos cidadãos comunicar à sua autarquia a existência de equipamentos obsoletos que tem que ser recolhidos, nomeadamente, resíduos volumosos, Monos. Da mesma forma permite pedir a remoção de resíduos verdes urbanos;
- **Orçamento Participativo** – Um serviço que permite às autarquias ter acesso à opinião dos cidadãos acerca dos projetos onde deverá ser empregue o orçamento público. O serviço permite a criação de sugestões e a consequente votação nas mesmas.
- **Registo de canídeos** – Esta funcionalidade permite registar animais de estimação, devidamente legalizados *a priori*;
- **Notícias** – As últimas notícias da região, publicadas pelo próprio Município/Junta de Freguesia;

- **Eventos** – Alertas e calendarização de eventos a decorrer ou agendados para um futuro próximo;
- **Contactos úteis** – Contactos de emergência de bombeiros, hospitais, policia, entre outros;
- **Ocorrências** – Um serviço com um mapa interativo que ajuda a criar ou verificar ocorrências na zona, com possibilidade de ver o seu estado e previsões das suas resoluções. Por exemplo, obras/trabalhos nas vias;
- **Informação Transportes** – Neste serviço o Município/Freguesia pode disponibilizar os horários dos diferentes transportes (Barco, Comboio, Autocarro, Avião, Urbano, Metro, Elétrico, Ascensor e Elevador).

Destas funcionalidades referidas acima, o utilizador é alertado por notificação, quando é criado um conteúdo desde notícias, eventos, ocorrências e comentários (SouCidadão, s.d.).

4.2 Arquitetura da aplicação

Na Figura 3 está apresentado o diagrama da arquitetura de aplicação. Como podemos verificar no diagrama a aplicação tem dois componentes principais, o *Drupal* e *Cordova*.

O *Drupal* é responsável por toda a gestão de *server-side* da aplicação. Gere as permissões dos seus utilizadores, os conteúdos criados, visualizações e módulos adicionados.

Enquanto o *Cordova*, é a parte de *client-side* de nossa aplicação. Esta *framework* transforma o código de *HTML, CSS* e *JS*, para uma aplicação *mobile*. Este processo envolve usar o *WebView* para mostrar o *HTML, CSS* e *JS* usando a *APIs* do *HTML*. A *WebView* comunica também com o "config.xml" para obter informações sobre a aplicação e os seus parâmetros.

Cordova também fica responsável por criar interface entre os *plugins* como a câmara ou notificações e as plataformas diferentes (Ex: *Android, iOS* e *WP*).

A conexão entre o *Cordova* e *Drupal* é feito através de *web-services* criados com *PHP*, que por sua vez esta conectada a uma base de dados de *cache*, onde a aplicação pode pedir as informações que necessita para não subcarregar o *Drupal*.

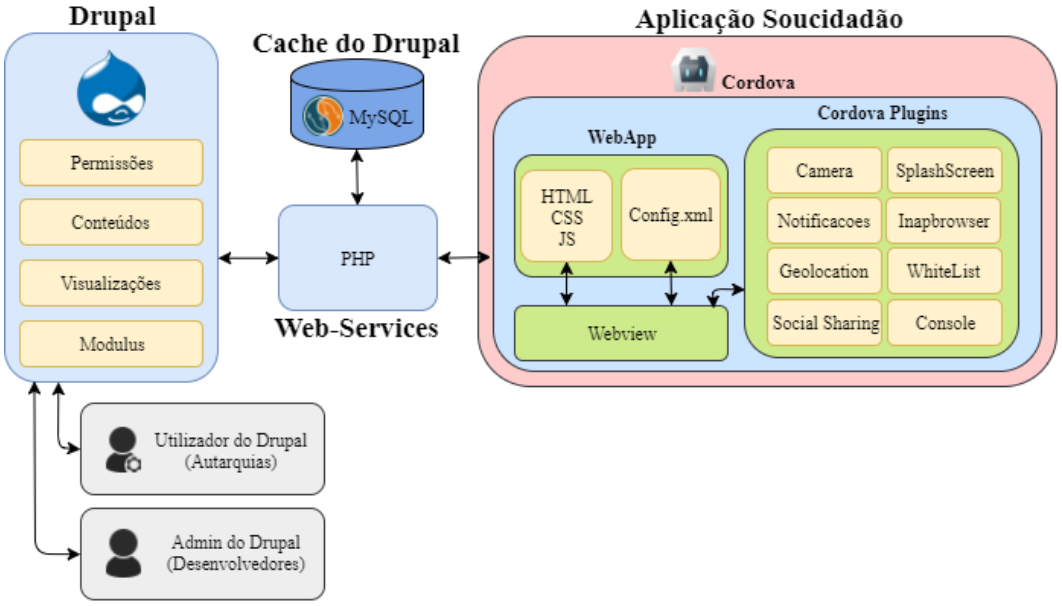


Figura 3 - Diagrama da arquitetura da aplicação de SouCidadão

4.2.1 Funcionamento de Drupal

O *back-end* de SouCidadão foi desenvolvida usando o *Drupal* como base. Nos próximos pontos é descrito o funcionamento de cada ferramenta do *Drupal* usada durante o estágio.

Vista de Autarquia

Apesar da plataforma de *Drupal* da Compta ser usada para os administradores gerirem o SouCidadão, também é uma plataforma onde as autarquias podem criar, ler e editar o seu conteúdo. Cada autarquia tem a sua autentificação para acederem à plataforma, podendo criar conteúdos como referendo local, criar notícias e gerir a sua plataforma no geral. A Figura 4 ilustra um exemplo de vista de Autarquia.

Figura 4 - Um exemplo de vista de uma autarquia no *Drupal*.

Tipos de Conteúdo

Todo o conteúdo criado no *Drupal* é chamado "nodes". Este conteúdo tem um comportamento semelhante a uma tabela na base de dados. O conteúdo tem o nome para distinguir de outros conteúdos, a sua chave primária (*nid*) e os campos que o programador adiciona.

Para adicionar um novo tipo de conteúdo, no *website* de *Drupal* da Compta, basta entrar em "Estruturas", escolhemos a opção de "Tipos de Conteúdos" e clicamos no "Adicionar tipo de conteúdo". De seguida surge um formulário que contém os campos "Nome" e "Descrição", sendo o "Nome" um campo obrigatório. Após a criação de conteúdo passamos para o próximo ecrã onde preenchemos os campos de conteúdo.

Para adicionar um campo basta adicionar um nome, escolher o tipo de conteúdo (Ex: texto, imagem, data, entre outros) e finalmente o tipo de *widget* para este campo (Ex: Imagem, Lista de seleção, Campo de texto, entre outros). Um exemplo de um tipo de conteúdo com nome de "Referendo" está apresentado na Figura 5.

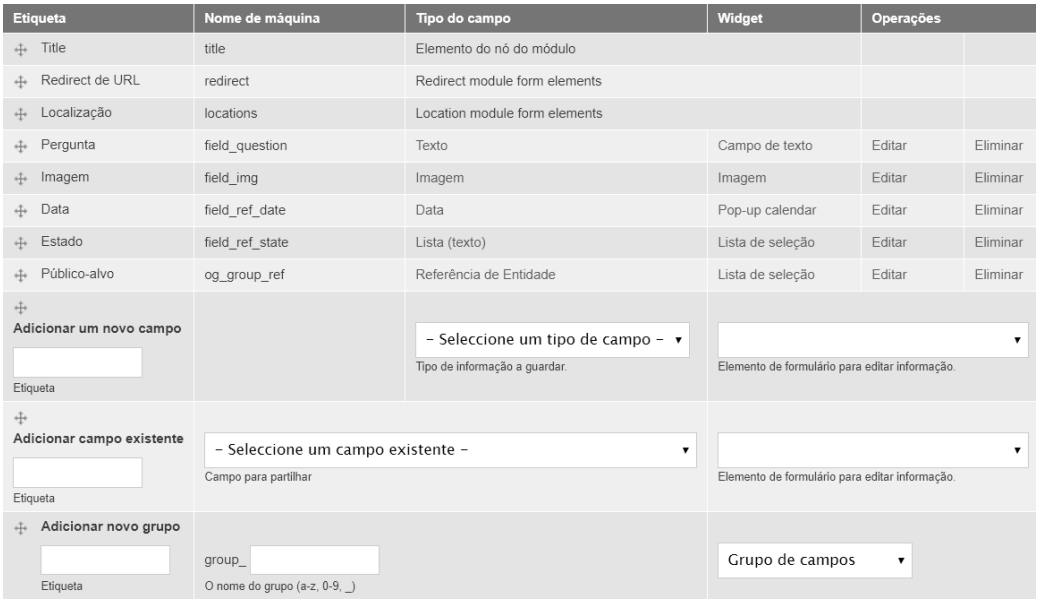


Figura 5 - Um exemplo de Tipo de Conteúdo no *Drupal*, neste caso o conteúdo com nome de "Referendo"

Visualizações

Uma visualização permite aos administradores criar, gerir e mostrar a lista de conteúdos. Comporta-se como as *views* de bases de dados. No decorrer do estágio, as visualizações foram usadas para dois fins: mostrar o conteúdo às autarquias de uma forma fácil de perceber (Vista de Autarquia) e enviar a informação dos conteúdos para os *web-services* que por sua vez vão processá-la e enviar para dispositivos móveis.

Para criar uma visualização, na página de "Estrutura", escolhemos a opção de "Visualizações" e carregamos a opção de "Adicionar nova visualização". De seguida preenchemos no formulário o campo "nome de visualização". Para maior facilidade de leitura, seguimos um padrão no preenchimento deste campo, tendo o formato de "vw_[nome]" caso seja uma visualização de "Page" que é usada para mostrar o conteúdo à autarquia. O nome de visualização tem formato de "vw_rest_[nome]" caso seja um "Serviço" e tem a finalidade de enviar a informação para dispositivos moveis.

Depois do nome ser adicionado, passamos para o passo de escolha do tipo de conteúdo a que esta visualização pertence, e a ordenação a ser usada. De seguida escolhe-se se trata de uma visualização do tipo "Page" ou do tipo "Serviço". Caso se selecione "Page", é necessário atribuir um nome/título e um endereço/caminho (ex: /vw-[nome]).

Após a visualização ser criada, pode-se proceder à adição ou remoção dos campos que esta visualização vai mostrar. Também é possível editar o formato em que a visualização é apresentada (Ex: tabela, gráfico, entre outros), o número de conteúdos por página e outros aspetos. O exemplo de uma visualização de tipo de "Page" com nome Resposta Referendo está mostrada na Figura 6.

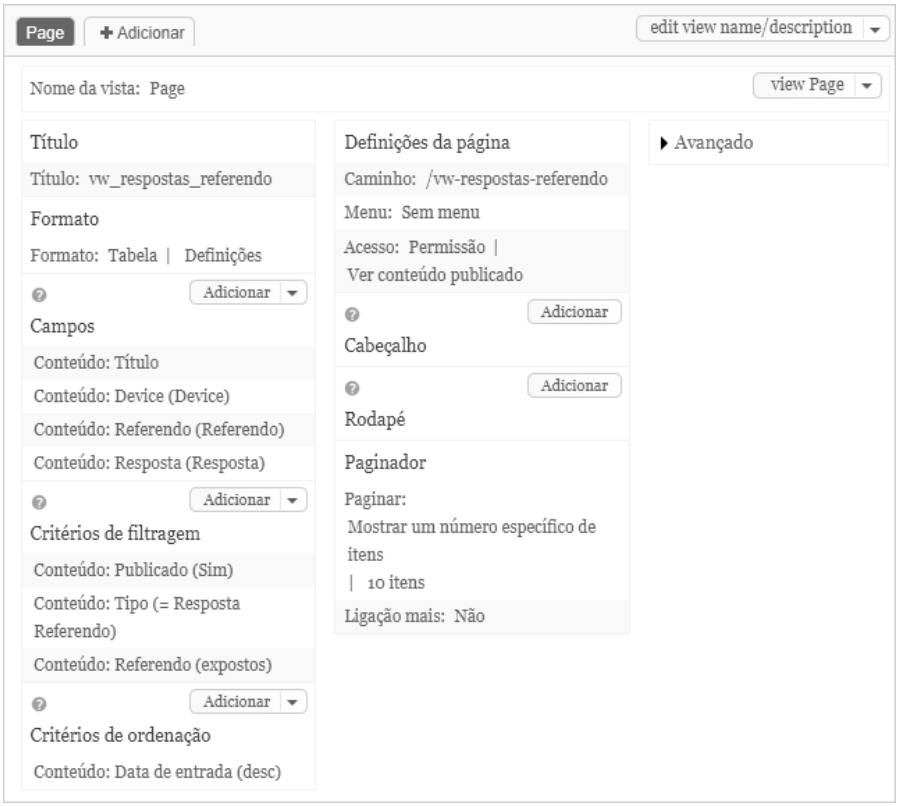


Figura 6 - Exemplo de visualização no *Drupal*, na qual se representa uma visualização de "Page" com o respetivo "Título", "Formato", "Endereço" e outros campos editáveis pelo administrador.

Regras

No *Drupal* as regras são semelhantes a *triggers* nas bases de dados, ou seja, são lançadas quando existe algum acontecimento (Ex: Criação ou edição de um tipo de conteúdo).

Para criar uma regra no *Drupal*, na página de "Configuração", escolhe-se a opção de "Regras" e carrega-se no botão de "Adicionar nova regra". No formulário apresentado, preenche-se o campo "*nome"* e escolhe-se o evento que vai lançar esta regra. Durante o desenvolvimento usam-se somente os eventos do Nó (tipo de Conteúdo), em especifico os eventos "Após guardar novo conteúdo", "Após atualizar um conteúdo existente" e "Após

apagar conteúdos". Caso seja necessário adicionar mais eventos a uma regra, basta editar o conteúdo. Após criar a regra e os seus eventos, a ação que é executada é escolhida no momento em que a regra dispara. Normalmente esta ação é uma função do *web-service* criada anteriormente.

Páginas

No *Drupal* o "Gestor das Páginas" permite aos administradores criar, editar e apagar as páginas. As páginas oferecem a vantagem de poderem ser personalizadas. Por exemplo, é possível adicionar múltiplos painéis que podem conter uma visualização, um código *HTML* estático, um gráfico ou outro conteúdo. Também é possível escolher um *layout* predefinido ou criar um *layout* personalizado, modificar o aspeto visuais adicionando as classes de *CSS*, e muitas outras funcionalidades. Podemos verificar um exemplo de uma página estática na Figura 7.

Figura 7 - Exemplo de uma página Serviço no *Drupal*, na qual se representam na vista de autarquia, os serviços que são oferecidos, esta página tem um *HTML* estático com *CSS* adicionado.

Menus

Durante o desenvolvimento, foi necessário adicionar novas ligações ao menu de Vista de Autarquia, em função dos novos serviços criados. Este processo engloba ir à página de estrutura e selecionar a opção de "Menus". Na lista apresentada, na linha de "Menu Lateral" escolhemos a opção de" Listar ligações", de seguida preenchemos os campos como nome e caminho para a página, guardando esta informação. É importante adicionar os *styles* a

ficheiro de *CSS* associado menu, para garantir que a ligação tem um aspeto semelhante a resto de menu. Um exemplo de menu pode ser verificado na Figura 4.

Web-Service

Web-Services são os ficheiros de *PHP*, que ficam com a responsabilidade de realizar as comunicações ente *front-end* e *back-end* da aplicação. Estes ficheiros estão localizados numa pasta do servidor. Cada serviço desenvolvido tem o seu próprio ficheiro *web-service*, que por sua vez tem um ficheiro de *log* associado, onde vão ser imprimidos os erros que podem vir a acontecer, facilitando o processo de *debugging* ao programador.

4.2.2 Notificações

Foi-nos proposto melhorar o sistema de notificações na aplicação, pois à altura a aplicação recebia notificações, mas não abria as páginas específicas. A nossa tarefa foi implementar uma funcionalidade de modo a que ao receber uma notificação, o conteúdo pudesse abrir na aplicação.

Após realizar esta tarefa, foi pedido o melhoramento do aspeto visual das notificações na plataforma de Android, em específico, adicionando um novo modo de representação de notificações chamado *large-format*. Este modo adiciona a funcionalidade de mostrar as notificações expandidas, tendo um título, uma descrição e uma imagem.

Para efetuar a primeira tarefa, foram feitas modificações ao *front-end* e *back-end* na aplicação. Como a Figura 8 mostra, as notificações só são enviadas quando uma das Regra de notificações é dispara. Neste momento existem quatro regras das notificações, no momento de criação de um conteúdo como um evento, uma notícia, ou um comentário de ocorrência, e também, no caso de o estado de ocorrência for modificado. Quando a regra é lançada, é executada a função "sendPush" de *web-service* chamado "WS_drupal_sendPush.php", que por sua vez acede a lista de dispositivos subscritos as notificações deste conteúdo e envia as notificações para os *smartphones*.

De seguida a aplicação recebe a notificação, e adiciona as listas de notificações, ficando a espera que o utilizador a abra. Quando o utilizador abre a notificação recebida, a aplicação abre a página de conteúdo de notificação.

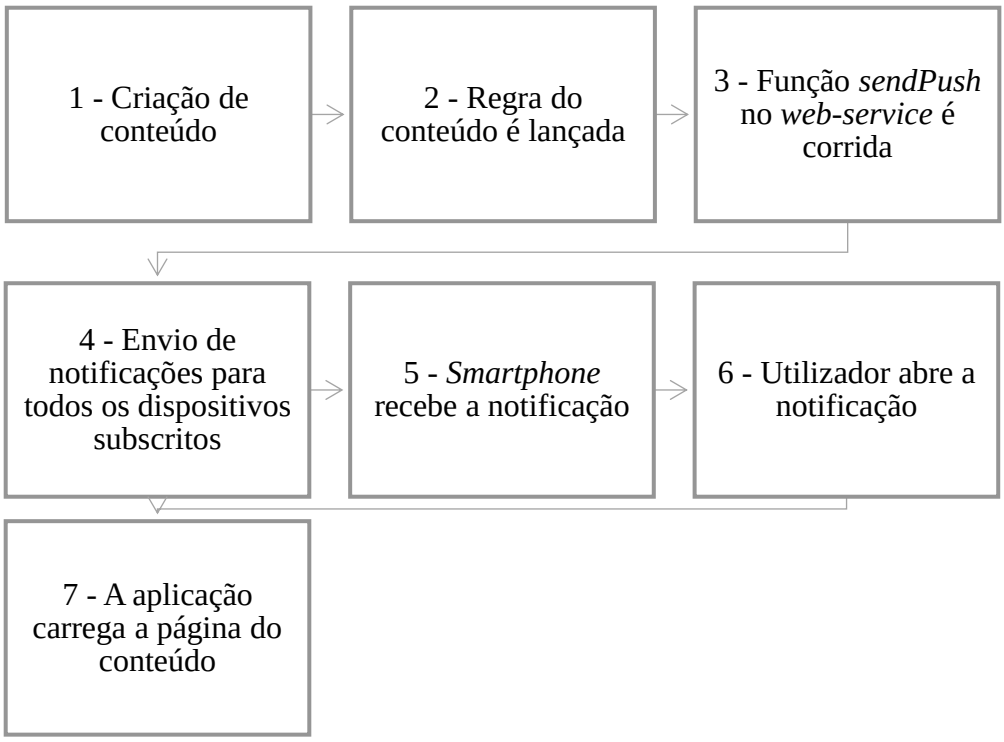


Figura 8 - Diagrama do processo de envio de notificação.

Back-End

De forma a podermos realizar as modificações descritas anteriormente, em primeiro lugar foi necessário planear quais as novas informações relativas às notificações vão ser enviadas. No momento de desenvolvimento, as notificações enviavam as informações apresentadas na Tabela 1, mas, no entanto, concluiu-se que se deveria alterar estas informações para as presentes na Tabela 2, sendo só o "pushid" usado para primeira tarefa, enquanto as informações como imagem e texto usadas para segunda tarefa de modificações visuais de notificação.

Tabela 1 - Lista de informações de notificação antes de modificações.

Variável	Descrição
nid	ID de Porto
texto	Nome do porto (Ex: Abrantes, Tomar).
tipo	Tipo de notificação (Ex: evento, noticia, ocorrência ou comentário de ocorrência)
titulo	Título de evento, noticia ou ocorrência

Para começar a implementação destas modificações, adicionando as novas informações da Tabela 2 no ficheiro "soap_tasks.js". Após de gravar o ficheiro, o *web-service* com nome de *sendPush* que se encontra no *Drupal* é atualizado, contendo os novos campos (*pushid* e imagem).

Tabela 2 - Novas informações de notificação

Variável	Descrição
nid	*ID* de Porto
pushid	*ID* de evento, noticia ou ocorrência
imagem	*URL* da imagem
texto	Descrição de evento, noticia, ocorrência ou comentário de ocorrência
titulo	Título de evento, noticia ou ocorrência
tipo	Tipo de notificação (Ex: evento, noticia, ocorrência ou comentário de ocorrência)

No próximo passo foram modificadas as regras do *Drupal* para cada tipo de conteúdo (evento, noticia, ocorrência ou comentário de ocorrência), associando, a informação de *pushid* a *id* de conteúdo, e imagem a imagem de conteúdo.

Em seguida foram efetuadas as modificações de *web-service* com nome de "WS_drupal_sendPush.php". Este *web-service* é responsável por envio das notificações para todos os dispositivos subscritos. Neste *web-service*, foram modificados alguns aspetos da função *sendPush* para processar os dados novos. Esta função é lançada quando a regra dispara.

Front-End

Na aplicação, o plugin que dá suporte às notificações é *PushPlugin*. Este plugin foi descontinuado no ano de 2014, consequentemente, isso provocou alguns problemas com *Windows Phone*, que infelizmente não foram possíveis de resolver, por sua vez isso impossibilitou a implementação desta funcionalidade no *Windows Phone*. Foi falado na possibilidade de atualizar para o novo plugin, mas esse processo implicaria o abandono da versão *WP8* e usar a versão mais recente, *WP10*. Por esse motivo foi decidido continuar o

uso do *plugin* corrente para abranger maior gama de telemóveis, abandonando a funcionalidade de abrir o conteúdo das notificações no *Windows Phone*.

No caso de *iOS*, os certificados de notificações não estavam atualizados na altura do desenvolvimento desta funcionalidade, impedindo a realização dos desenvolvimentos e testes necessários. No entanto, foi efetuado o desenvolvimento desta funcionalidade para *android*.

Como a aplicação já recebia as notificações, não foi necessário realizar muitas modificações no código-fonte. No entanto foi preciso modificar o ficheiro responsável pela receção de notificações e os ficheiros de serviços dos eventos, noticias e ocorrências.

Em cada ficheiro dos serviços, foram adicionadas uma ou mais funções, que usam as informações da notificação presentes na Tabela 2. Em específico, foi usada a variável *pushid*, para mostrar a página do conteúdo.

Ao finalizar as modificações anteriores e garantir que elas funcionam, procedeu-se à modificação do ficheiro responsável pela receção das notificações. Neste ficheiro foi modificada a função chamada *onNotificationGCM* que é responsável pela receção de notificações no *android*. Nesta função, usando a variável "tipo" da Tabela 2, construiu-se um "switch" que lança as funções dos serviços construídos anteriormente. Na conclusão destas modificações fez-se a verificação das notícias que estão a abrir o seu conteúdo na aplicação, tentando pelos possíveis erros.

4.2.2.1 Melhoramento do aspeto visual de Notificações

Após as a conclusão da primeira tarefa, procedeu-se ao desenvolvimento da segunda tarefa, que consistiu no melhoramento do aspeto visual das notificações. Foi fornecido um exemplo de *layout* para seguir, mostrado na Figura 9.

Figura 9 - Protótipo de *layout* das notificações "SouCidadão" em sistema Android.

Como falado anteriormente, o *plugin* das notificações foi descontinuado, consequentemente deixando de ter suporte para estas funcionalidades. Por este motivo foi necessário modificar o ficheiro *GCMIntentService.java* que faz parte de *PushPlugin*. Nesse ficheiro foi usada a *library android.support.v4.app.NotificationCompat* para adicionar as notificações o modo *large-format*.

Para garantir que não houvessem problemas, foram adicionadas as imagens por defeito às notificações, uma vez que poderiam existir casos de notificações que não tenham uma imagem associada (Ex: um evento poderia não ter imagem). Um dos problemas surgiu no recebimento de notificações, que sem permissão prévia se abriam automaticamente. Essa situação foi resolvida, garantindo que as notificações recebidas ficariam em modo de espera até que o utilizador a abrisse.

A Figura 10 mostra o aspeto final das notificações, em formato normal e formato grande.

a) b)

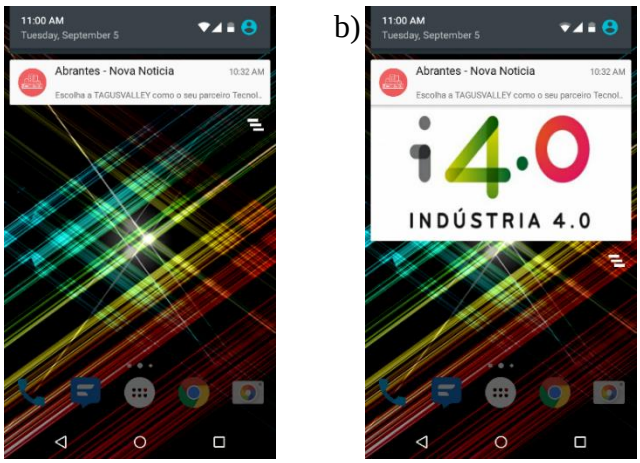


Figura 10 - Resultado final das alterações na apresentação de notificações:
a) Exemplo de uma notificação em modo normal;

b) Exemplo de uma notificação em modo *large-format*.

4.2.3 Serviço Recolha Monos e Verdes

O serviço de Recolha Monos e Verdes (RMV), é parte integrante da segunda tarefa desenvolvida no SouCidadão após as notificações. RMV é um serviço que permite aos cidadãos comunicar ao governo local a existência de equipamentos obsoletos (Ex: frigoríficos, maquinas de lavar, etc.) ou resíduos verdes que têm de ser recolhidos.

Para efetuar estas implementações, o primeiro passo foi começar por adicionar o conteúdo necessário no *back-end* da aplicação.

Back-End

Começando por adicionar no *Drupal* um novo "Tipo de Conteúdo" chamado de "Recolha MV", preenchendo com informações representadas na Tabela 3.

Tabela 3 - Variáveis de "Tipo de Conteúdo" do RMV

Variável	Tipo de campo	Widget	Descrição
name_cidadao	Texto	Campo de texto	Nome cidadão
contact_number	Inteiro	Campo de texto	Numero de contacto
nif_cidadao	Inteiro	Campo de texto	NIF
pikingtype	Texto	Campo de texto	Tipo de recolha
pikingmaterials	Texto	Campo de texto	Materiais
pikingdate	Texto	Campo de texto	Data
pikingaddress	Texto	Campo de texto	Localização
email	Texto	Campo de texto	Email
observations	Texto	Campo de texto	Observações
imagem_mv	Imagem	Imagem	Imagem
og_group_ref	Referência de entidade (Porto)	Lista de seleção	Público-alvo

Após da criação do "Tipo de Conteúdo", procedeu-se à criação da sua visualização. Esta visualização será do tipo "Page", pois é usada somente para que as autarquias visualizem a lista dos RMV existentes. Na visualização criada, teve-se o cuidado de

modificar os cabeçalhos da tabela para uma leitura mais fácil. Com próximo passo, criou-se uma "Página" com endereço/caminho de "/recolhas_mv". Nesta "Página" foi adicionado um título e a visualização criada anteriormente, a Figura 11 mostra o aspeto visual final de RMV, no ponto de vista de uma autarquia. A seguir foi modificado o menu para criar uma ligação com a página.

Figura 11 - Exemplo de uma lista de RMV de ponto de vista de uma autarquia.

Para o *client-side* da aplicação ter a capacidade de comunicar com o *Drupal*, foi criado um serviço do RMV chamado "WS_drupal_RMV.php". Este *web-service* contém uma função com nome de "AddNewRMV". Esta função é responsável por receber as informações que a *client-side*, ou seja as informações que utilizador envia, comunicando com o *Drupal* para adicionar um tipo de conteúdo RMV. Por último, é enviada a resposta a *client-side* se foi possível ou não adicionar o conteúdo.

Front-End

No *front-end*, o desenvolvimento começou pela criação do ficheiro com nome "appLA_showScreenRMV.js", que ficaria responsável pelo serviço de RMV. Nesta tarefa foi pedido a criação de duas janelas, uma responsável por selecionar o tipo de recolha, como mostrado na Figura 12 e outra por preencher o formulário de recolha mostrado na Figura 13.

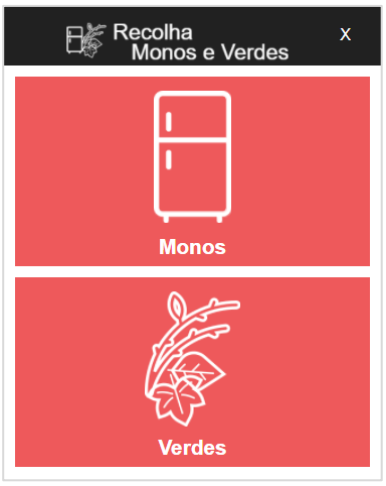
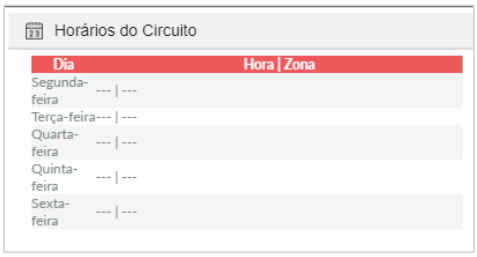


Figura 12 - Seleção de recolha dos monos ou verdes no SouCidadão

A segunda janela está separada em dois componentes, a tabela de informações, como é mostrado na Figura 13-a), e o formulário de recolha como é mostrado na Figura 13-b).

a)



b)

Figura 13 - Segundo ecrã de serviço de recolha dos monos ou verdes no SouCidadão:
a) Horário do Circuito;
b) Formulário de serviço de recolha dos monos ou verdes.

A tabela de informações dos Horários de Circuitos será preenchida pela autarquia.

No caso de formulário, depois de ser criado foi necessário ter alguns cuidados, tais como as validações dos campos ou o funcionamento correto do campo de data em todas plataformas (*Android, WP 8 e iOS*).

São usadas validações de tipo de expressões *Regular Expression (RegExp)* para validar os campos como nome, email, telemóvel, NIF e data, validar se os campos obrigatórios contêm informação, validar se o tipo de ficheiro é uma imagem (*png, jpg, bmp e gif*) e se o utilizador aceita os termos.

Para facilitar o uso da aplicação aos utilizadores, algumas informações inseridas anteriormente (nome, email, NIF e telemóvel) podem ser guardadas no *localhost*, para o utilizador evitar preenche-las de novo. No entanto existe um botão para que possam ser apagadas.

Como o objetivo é enviar este formulário à *Drupal*, foi criada uma função chamada "submitRMV", que depois de validar os dados, envia-os ao *web-service* de RMV (WS_drupal_RMV.php). Após enviar os dados fica à espera da resposta do *web-service*, que poderá informar se o processo decorreu com sucesso ou houve um erro.

4.2.4 Serviço Referendo Local

O objetivo de serviço de Referendo Local, foi criar uma plataforma onde seja possível ao governo local colocar perguntas a utilizadores, que por sua vez podem votar como "Sim" ou "Não". Estas questões geram as estatísticas de opinião, que a autarquia pode usar para melhorar a cidade ou para outros fins.

Para implementar este serviço, procedemos as modificações necessárias no *back-end* da aplicação.

Back-End

Como este serviço tem o referendo e a resposta do mesmo, faz sentido adicionar dois tipos de conteúdos no *Drupal*. O nome dado a primeiro tipo de conteúdo é "Referendo", que é preenchendo com as informações da Tabela 4.

No conteúdo da Resposta do Referendo, ou seja, o segundo conteúdo, foi dado o nome de "Resposta Referendo" que por sua vez será preenchido com as informações da Tabela 5.

Tabela 4 - Variáveis de “Tipo de Conteúdo” do Referendo

Variável	Tipo de campo	Widget	Descrição
question	Texto	Campo de texto	Questão
img	Imagem	Imagem	Imagem
ref_date	Data	Pop-up calendar	Data do inicio e fim
ref_state	Lista (Texto)	Lista de seleção	Estado (Ativo ou Desativo)
og_group_ref	Referência de entidade (Porto)	Lista de seleção	Público-alvo

Tabela 5 - Variáveis de “Tipo de Conteúdo” do Resposta Referendo

Variável	Tipo de campo	Widget	Descrição
answer	Booleano	Caixas de seleção	Nome cidadão
device_id	Inteiro	Campo de texto	Numero de contacto
referendo	Referência de entidade (Referendo)	Lista de seleção	Público-alvo
og_group_ref	Referência de entidade (Porto)	Lista de seleção	Público-alvo

Após de adição destes conteúdos, avançou-se para as visualizações.

Para criar uma “Vista de Autarquia” mais apresentável, foi necessário criar três visualizações de tipo “Page”, a visualização de lista de Referendos, “vw_referendos”, a visualização com a lista de Resposta Respostas, “vw_respostas_referendo” e por final uma visualização de formato *chart* para criar um *piechart* que representa a percentagem de respostas positivas e negativas, chamado “graf-resp_referendo”. Na visualização de Referendo usamos um módulo de *Drupal* chamado “ColorBox” que permite nos criar um tipo de janela para mostrar os detalhes do referendo, mais especificamente, as Respostas Referendo e o seu *piechart*. Esta *ColorBox*, que mostra os detalhes de Referendo Local, pode ser verificada na Figura 15.

Em seguida procedemos a criação das “Páginas” no *Drupal*. Atribuímos o nome de “Referendo Local” a “Página” de conteúdo de Referendo com o caminho de “/referendo”,

em seguida, adicionamos os conteúdos da mesma, o botão para as autarquias adicionarem um novo conteúdo de tipo Referendo Local, o título da página e a visualização "vw_referendos". Podemos verificar o resultado final na Figura 14.

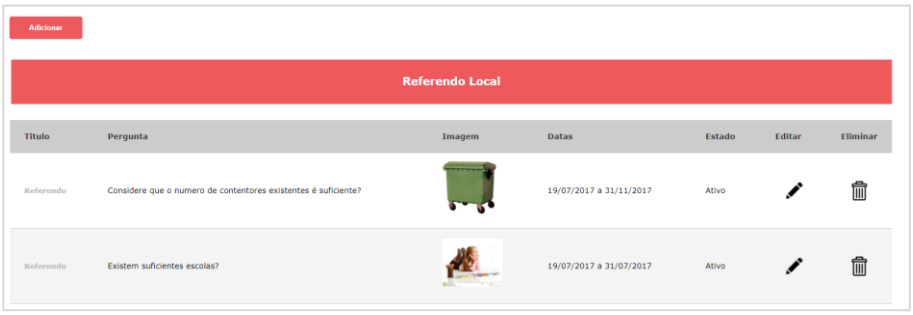


Figura 14 - Exemplo de uma lista de Referendos de ponto de vista de uma autarquia.

Em seguida procedeu-se à criação da "Página" de Resposta Referendo, que vai ser usada para mostrar os detalhes de Referendo Local. Nesta página adicionámos a visualização de Resposta Referendo e o seu *piechart*. Um exemplo de detalhes de um referendo pode ser verificado na Figura 15.

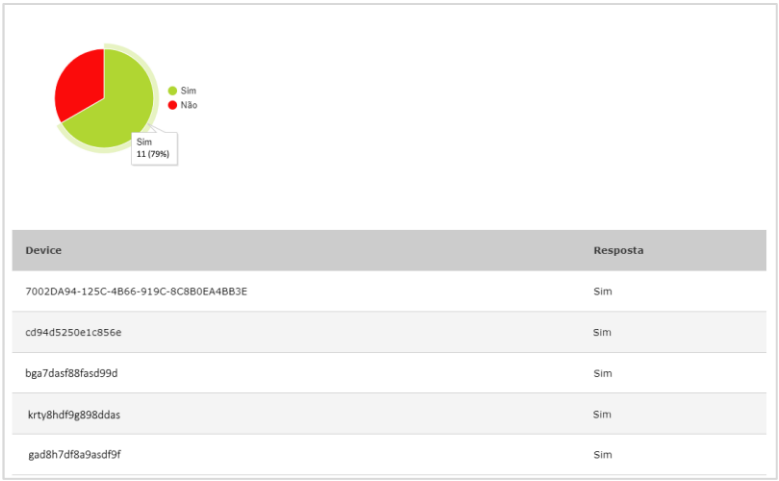


Figura 15 - Exemplo de detalhes de um Referendo de ponto de vista de uma autarquia.

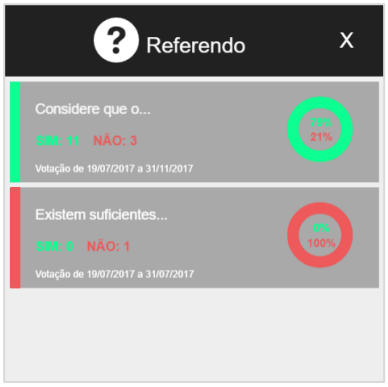
Após a conclusão de criação de visualizações do tipo "Page", procedeu-se à criação de visualizações de "Serviços". Estas visualizações servem para enviar as informações para o *web-service* do Referendo Local. Serão necessárias duas visualizações, uma visualização de tipo de Referendo chamada "vw_rest_referendos" e outra de tipo Resposta Referendo com nome de "vw_rest_resposta_referendo". Após a criação das visualizações, foram criados os *web-service* que as vão usar.

Foi dado o nome de “WS_drupal_question.php” ao novo *web-service*, no qual foram criadas três funções. A função “GetQuestions” que fica responsável por envio da lista dos Referendos e as respostas dos mesmos para o client-side, a função “GetDeviceVoted” que verifica se o utilizador já votou na questão e finalmente “AddNewResponce”, que fica com responsabilidade de adicionar os novos votos ou mudar os votos antigos caso o utilizador deseja. Depois de terminado o desenvolvimento no *back-end*, avançou-se para o desenvolvimento de *front-end*.

Front-End

Este processo engloba criar dois ecrãs, o primeiro ecrã representado na Figura 16-b), mostra a lista de todos os Referendos Locais da autarquia, e o segundo ecrã representado na Figura 16-b) mostra um ecrã de votação, em qual esta presente a questão completa, a imagem e os botões de voto.

a)



b)

Figura 16 - Ecrãs do serviço do Referendo Local
a) A lista de Referendos Locais na aplicação *mobile* de SouCidadão
b) Ecrã de detalhes de um Referendo Local na aplicação *mobile* de SouCidadão

No primeiro ecrã é usada a função de “GetQuestions” do *web-service* de Referendo Local para carregar os dados de todos os referendos e as suas estatísticas. Foi tido o cuidado de desligar a votação, em casos onde a questão já não esteja ativa ou que o período de votação já tinha terminado. Neste caso os referendos com o estado desativo permanecem na lista para o utilizador ter a possibilidade de verificar a opinião de outros cidadãos ou verificar o seu voto nos detalhes do referendo.

O segundo ecrã mostra uma vista detalhada do referendo, as informações de qual são passadas do primeiro ecrã para não subcarregar o servidor com mais medidos. Foi usada a função "GetDeviceVoted" para verificar se o utilizador já votou, e no caso de ter votado, o voto anterior fica marcado. No caso de o Referendo Local estar com estado "ativo" e o período de votação esta dentro das margens, o utilizador pode também votar ou mudar o seu voto caso já tenha votado anteriormente, para essa ação usamos a função de "AddNewResponce". Após a terminar o desenvolvimento deste serviço foram realizados os testes para garantir um bom funcionamento do serviço.

4.2.5 Outros Desenvolvimentos

O Serviço Informação Transportes teve como foco, criar uma lista de horários de transportes. Este serviço foi desenvolvido juntamente com uma estagiária nova na empresa. A minha função foi explicar o funcionamento de *back-end*, ou seja, o *Drupal* e os *web-services* e ajudar nas partes mais complicadas no *front-end* de aplicação.

4.2.6 Cache

Os serviços como Referendo Local e Informação Transportes podem precisar de receber muita informação do servidor, por esse motivo foi implementado um sistema de *cache* semelhante a sistema de *cache* de outros serviços.

Para implementar estas funcionalidades começamos por adicionar as tabelas na base de dados que ficam a guardar os dados da *cache*. A Tabela 6 mostra as informações que uma tabela de *cache* guarda nas bases de dados. A informação que é guardada nessas tabelas, é a resposta das funções que enviam os dados para o *front-end* da aplicação, ou seja, nos casos de Referendo Local a resposta de função "GetQuestions" e no caso de Informação Transportes a resposta de função "GetTransportes".

Tabela 6 - Uma tabela do *cache* no SouCidadão

Variável	Tipo de campo	Descrição
Id	int	*Id* da tabela
nid	int	*Id* do Porto (Ex: Tomar, Abrantes)
last_updated	datetime	A data de ultimo *update* de campo
json	longtext	A cache em formato de *json*

Depois de tabelas de bases de dados adicionas nos *web-services*, modificamos as funções "GetQuestions" e "GetTransportes", para verificarem se existe *cache* na base de dados, caso ela exista, na resposta da função, enviamos as informações da *cache*, caso não exista ou retorna um erro, o resto da função é executada e o resultado é enviado para utilizador.

Para os passos referidos acima funcionarem, também é necessário modificar o *web-service* com nome de "WS_drupal_cache.php", que é o ficheiro responsável pela adição ou renovação da *cache* na base de dados. Foram adicionadas duas funções a este *web-service*, "UpdateReferendo" e "UpdateTransportes". Quando uma dessas funções é chamada, vai executar a função "GetQuestions" ou "GetTransportes", para pedir os dados mais recentes dos serviços.

Após da implementação dessas funcionalidades, foram adicionadas as regras de "cache_referendo", "cache_resposta_referendopara" e "cache_transportes", que irão ser lançadas ao adicionar, modificar ou apagar um tipo de conteúdo de Referendo Local ou Informação Transportes. Com esses passos concluídos, foram realizados os últimos testes para garantir que a *cache* está a funcionar.

4.2.7 Testes

Durante o desenvolvimento, foi necessário realizar inúmeros testes para verificar se a aplicação funciona corretamente. Os testes iniciais, foram feitos pela emulação de *android*, normalmente usando o *browser* do Chrome. Durante estes testes foi necessário garantir que a aplicação suporta uma grande gama de resoluções, e com o uso de uma emulação é possível verificar, em tempo real, se existem erros de *layout* ou *style* na aplicação.

No entanto, a emulação no *browser* não garante o funcionamento correto nos dispositivos físicos, por esse motivo é essencial fazer estes testes em cada plataforma móvel, usando *smartphones* diferentes. Como as aplicações são primariamente desenvolvidas para a*ndroid*, existe sempre um número maior de problemas nos sistemas de *iOS* e *Windows Phone* para corrigir. Após a realização destes testes, garantindo que não existem mais erros, pode-se passar para o próximo passo de desenvolvimento, de acordo com o exposto no capítulo 3.

5 PaxVoice

Este capítulo apresenta a plataforma PaxVoice e o trabalho desenvolvido neste projeto no âmbito do estágio.

5.1 Introdução

PaxVoice é uma plataforma social lançada em 2015, com objetivo de recolher e divulgar a opinião pública mundialmente sobre um dado tópico, em tempo real. No PaxVoice, como o nome indica, a parte mais importante da plataforma é a "Voice", ou seja, o utilizador.

O utilizador, depois de se registar, pode criar uma "Voice", isto é, uma pergunta, comentário ou opinião, com a qual o resto dos utilizadores podem interagir. Podendo assim, votar positiva ou negativamente e/ou adicionar comentários. Através das interações dos utilizadores é possível gerar estatísticas de respostas que são divulgadas em tempo real a todos os utilizadores.

PaxVoice neste momento está disponível para *smartphones* dotados de sistema operativo *Android*, *WP* e *iOS*, estando ainda disponível em formato web.

A aplicação esta integrada com o *Facebook* e *Twitter*. O *Facebook*, neste momento, é o único meio de autentificação na aplicação, no entanto, o sistema de autentificação por e-mail foi desenvolvido durante o decurso do estágio e será implementado na próxima versão. No caso de *Twitter*, permite criar "Voice" automaticamente através de um *tweet*, bastando acrescentar o ***#paxvoice*** (Compta, 2017).

5.2 Arquitetura da aplicação

A Figura 17 apresenta a arquitetura da aplicação PaxVoice desenvolvida para web.

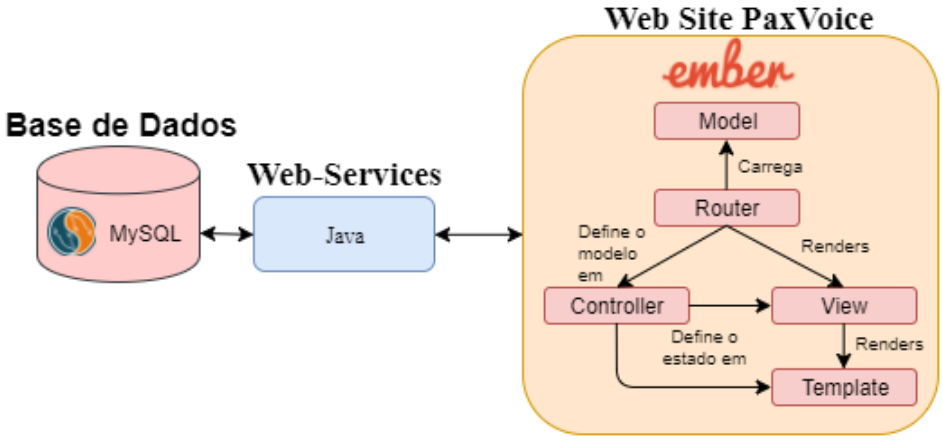


Figura 17 - Arquitetura do PaxVoice

O *front-end* do PaxVoice foi desenvolvido em *Ember.js*, que é separado em próximos componentes, a *Model*, *Router*, *Controller*, *Veiws* e *Template*.

A aplicação começa pelo *Router* que carrega as informações de *Model*, e passa estas para o *Controller*, este por sua vez, define os estados em *Views* e *Templates*. O *Router* também faz o render da *View*, que por sua vez faz o render do *Template*.

Usamos os *web-services* criados com *Java* para comunicar com o servidor e bases de dados.

A aplicação também tem uma parte *mobile*, que não esta representada na arquitetura, pois o trabalho desenvolvido não é suficiente para a adicionar.

5.3 Desenvolvimento

Como a primeira tarefa de estágio, foi me proposto corrigir os erros nas plataformas de *mobile*. Como por exemplo o *logout* não funcionava em alguns casos, as imagens de formato *SVG* nem sempre apareciam ou problemas visuais com *CSS*. A aplicação *mobile* foi desenvolvida com a *framework Ionic*, que é construído por cima de *Angular.js*, ou seja, o uso de *Ionic* é muito similar ou até igual a *Angular.js*, retirando a componente visual. Estas tarefas serviram como um método de aprendizagem de funcionamento da aplicação para futuros desenvolvimentos.

A segunda tarefa, que é a tarefa mais importante atribuída no Paxvoice, foi o desenvolvimento do *front-end* de um sistema de autentificação por email, na parte de *website*.

Durante o desenvolvimento de sistema de autentificação, foram-me fornecidos os *layouts* para usar como referência no desenvolvimento. Estes *layouts* foram criados com um *website InVision*.

Nos pontos seguintes apresenta-se o código desenvolvido, no entanto pormenores como variáveis e algumas funções gerais não serão aprofundados, por não se tratar de conteúdo relevante ao relatório, tornando-o demasiado extenso e exaustivo.

5.3.1 Resources

Antes de começar os desenvolvimentos, foram criados os Resources no ficheiro de "routers.js". Estes servem para definir as *Routes* dos conteúdos a serem usados no futuro, representados na Tabela 7, com as suas descrições.

A criar uma *Resource*, é automaticamente criado o *controller*, a *route* e o *template*. Estes seguem a mesma política, ficando o *controller* com nome [name]*Controller*, a *route* com nome de [name]*Route* e o *template* com nome de [name], onde o "name" é o nome que damos a *Resource*. Como exemplo se o nome de *Resource* fosse *index*, ficávamos com *IndexController*, *IndexRoute* e *index* (nome do *template*).

Tabela 7 - Tabela com os *Resources* criados no ficheiro *routers.js*

Nome do Resource	Descrição da página
loginEmail	Login por email
registerEmail	Registar com email
checkEmail	Ativação de conta (*link* do email)
requestRecoverPass	Pedido de recuperação de password
recoverPass	Página de modificação de password (*link* do email)
askValidation	Pedido de reenvio de email de ativação
forgotPassword	Recuperação de password (*link* do email)
settings	Propriedades de conta de utilizador

5.3.2 Sistema de Autentificação

Para criar um sistema de autentificação por email, usamos o *plugin* "ember-simple-auth", que já está a ser utilizado na autentificação por *Facebook*. Modificou-se o ficheiro "app.js", que guarda as configurações do *website* e de sistema de autentificação, adicionando a autentificação por email, seguindo o exemplo do *Facebook*.

Este sistema de autentificação é separado em 3 partes, "authenticate", "restore" e "invalidate".

Authenticate

No "authenticate", ou seja, autentificar, é usada a *API* "/users?token=", que envia o *token* como parâmetro, sendo o *token* a juncão email e password no formato encriptado. Esta *API* verifica se o utilizador existe, no caso de existir, cria um objeto de sessão, onde vai guardar as informações pessoais do utilizador, também guardando o tipo de login que o utilizador realizou, ou seja, se foi por email ou por *Facebook*, pois, depois de login realizado, a representação visual da informação do utilizador é tratada de forma diferente, conforme o tipo de login. Por exemplo, no caso de o utilizador realizar o *login* com *Facebook*, é usado como avatar o *URL* da imagem de *profile* de *Facebook*, enquanto ao realizar o *login* com email, a imagem é carregada do servidor.

Restore

O "restore", é usado para restaurar uma sessão. Esta restauração é usada por exemplo, quando o utilizador fechou a *tab* ou o *browser* sem fazer *logout*, e ao abrir o *website* de PaxVoice novamente, o *login* vai ser feito automaticamente.

Invalidate

No caso de "invalidate", é usado simplesmente para fazer o *logout*, ou seja, apagar o objeto de sessão.

5.3.3 APIs usados

Similarmente foram fornecidos os *APIs* necessários para os desenvolvimentos de todas as funcionalidades de sistema de autentificação por email, mostrados na Tabela 8.

Tabela 8 - Os *APIs* usados no desenvolvimento de sistema de autentificação

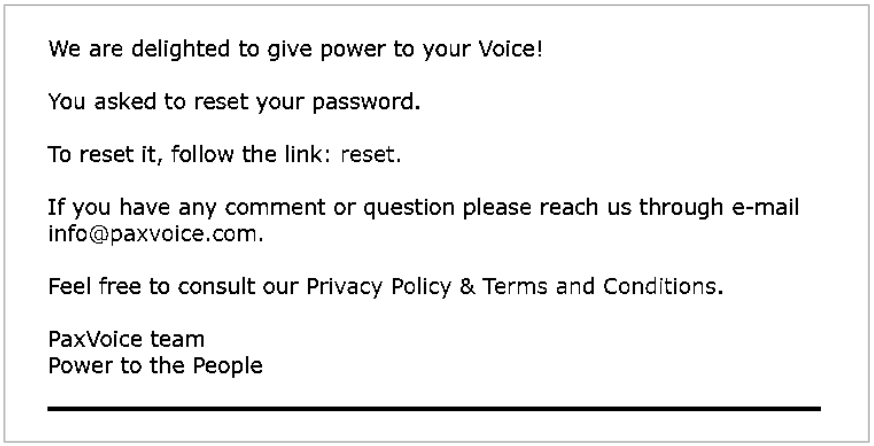
API	Tipo	Descrição
/users?token=[token]	GET	Devolve as informações pessoais do utilizador, usado na autentificação, onde o "token*"* é a juncão de email e *password* no estado criptografado.
/facebook/countries?locale=en_US&q=	GET	Devolve a lista de países.
/facebook/regions?locale=en_US &country=[country]&q=	GET	Devolve a lista dos distritos, onde o "country*"* é o país que o utilizador selecionou.
/facebook/cities?locale=en_US&country=[country]&&q=	GET	Usado no *Helper* "type-ahead-city", onde o "country*"* é o país que o utilizador selecionou.
/users/email?email=[email]	GET	Verifica se o email já existe no sistema (*true*) ou não (*false*), sendo o "email" é o email do utilizador.
/users/register?token=[token]	POST	Submete as informações de utilizador a base de dados, onde o "token*"* é a juncão de email e *password* no estado criptografado.
/settings?token=[token]	GET	Devolve as informações pessoais do utilizador na pagina de *Settings*, onde o "token*"* é a juncão de email e *password* no estado criptografado.
/settings?token=[token]	PUT	Submete as informações atualizadas de utilizador a base de dados, onde o "token*"* é a juncão de email e *password* no estado criptografado.
/settings/password?token=[oldToken]&newtoken=[newToken]	PUT	Modifica a *password* do utilizador, substituindo o *token* na base de dados, onde os "oldToken" e "newToken" são as juncões de email e *password* no estado criptografado.
/users/validation?token=[token]	GET	Usado apara ativar a conta depois de registar-se, onde o "token*"* é a juncão de email e *password* no estado criptografado.

API	Tipo	Descrição
/users/email?email=[email]	GET	Usado para pedir a recuperação da *password*.
/users/resetvalidation?token=[token]	GET	Usado para mudar a *password*, onde o "token*"* é a juncão de email e a *password* nova no estado criptografado.
/users/askValidation?email=[email]	GET	Pedido para envio de novo email de ativação de conta.

5.3.4 Confirmações por email

Existem algumas situações em que é necessário enviar emails de confirmação. Um email de confirmação tem como objetivo enviar um *link* ou informações ao utilizador. No caso de ser *link*, no que toca à autentificação por email, servem por exemplo, para confirmar o seu registo no sistema, ou para mudar a *password*.

Em específico, existem três tipos de emails. O email de confirmação de registar, o email de reenvio de confirmação de registar (o utilizador tem que fazer o pedido do mesmo no website) e finalmente o email de recuperação da *password*, que o utilizador pede se não consegue lembrar-se da *password*. Na Figura 18 esta presente um exemplo de email recebido.


We are delighted to give power to your Voice!

You asked to reset your password.

To reset it, follow the link: reset.

If you have any comment or question please reach us through e-mail info@paxvoice.com.

Feel free to consult our Privacy Policy & Terms and Conditions.

PaxVoice team
Power to the People

Figura 18 - Email de confirmação de pedido de mudança de *password*

5.3.5 Helpers

Um *Helper* é um elemento de *HTML* modificado, para possuir funcionalidades extras, alem das funcionalidades oferecidas pelo *Ember.js*, durante o desenvolvimento foram criados três *Helpers*.

Type-Ahead-City

Este *Helper* é do tipo *input*, usado para auto completar as cidades de um país, ou seja, ao escrever três carateres neste input vão nos ser mostradas as opções possíveis de completação, que pode ser verificado na Figura 19. Este auto complete usa a *API* "/facebook/cities?locale=[en/pt]&country=[país selecionado]&q=[dados do input]" para buscar as respostas.

Figura 19 - Exemplo de auto completação do campo de Cidade

avatarUpload

O "avatarUpload", é um *Helper* de tipo *input file*, que faz o *upload* do *avatar* do utilizador, para o fazer usa o ficheiro de "upload.php", que se situa no servidor.

date-picker

O "date-picker" é um campo de tipo *input*, é usado para, ao clicar no campo, ser apresentado um calendário, onde o utilizador seleciona a data do seu nascimento. Na Figura 20 está presente um exemplo de aspeto visual final da input de seleção de dia do nascimento.

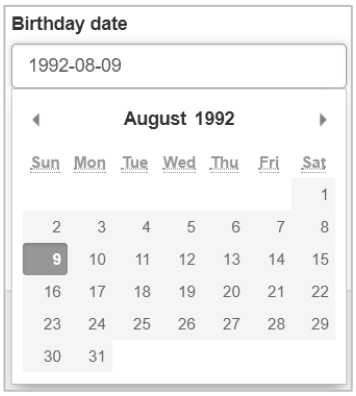


Figura 20 - Exemplo de um *Helper* com nome de “date-picker”

5.3.6 Registar

A página Registar é usada para registar um utilizador novo na base de dados.

Routes e Models

Uma *Route* indica o funcionamento de uma página, como por exemplo o seu nome, o seu endereço ou as suas informações. Uma *Model* no *Ember.js* é um objeto da página, que guarda as informações.

No *Route* de Registar, indicamos que o *template* de Registar vai realizar o *render* no *outlet* de “home”.

No seu *Model* vai ter as informações *default* de registar, ou seja, vai limpar os campos de página, substituindo com os valores de *Model* ao carregar da página.

Template

A página de registar será composta por duas partes, a primeira parte é usada para validar o email e a *password* como mostrado na Figura 21-a), e a segunda parte para adicionar e validar as informações pessoais de utilizador, como o seu avatar, o nome, sexo, a data de nascimento, país, distrito e a cidade como mostrado na Figura 21-b).

a) b)

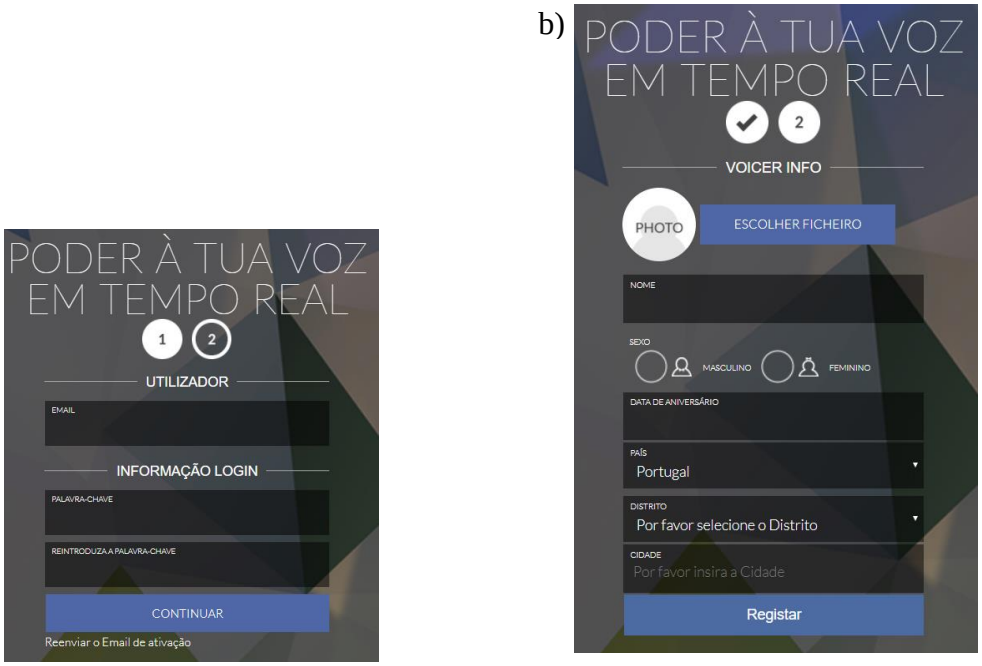


Figura 21 – Os ecrãs de registar no PaxVoice
a) Primeiro ecrã de registar no PaxVoice
b) Segundo ecrã de registar no PaxVoice

Controller

Como o primeiro passo, foram criadas as funções para validar os campos de registar, a função chamada "validateStepOne" valida o primeiro ecrã, em específico o campo de email e *password*, também verificando se a *pasword* e a reintrodução de *password* são iguais. No caso do email, também é verificado se o email já está a ser utilizado, ou seja, se o utilizador já tem a conta criada ou o email pertence a outro utilizador, esta verificação usa a *API* "/users/email?email=[email]", que retorna *false* caso seja possível criar a conta e *true* se o email já está em uso. Na validação do segundo ecrã, foi criada a função "validateSteptwo", que verifica se os campos obrigatórios como o país e nome estão preenchidos.

Após a criação das funções de validações, foram desenvolvidas as funções para carregar as informações dos países, distritos e cidades. Os países são carregados no abrir a página, usando a *API* de "/facebook/countries?locale=en_US&q=", os distritos são carregados após a escolha do país usando a *API* de "/facebook/regions?locale=en_US&country=[país]&q=", onde o campo país é o escolhido anteriormente, e nas cidades, foi usado o *Helper* chamado "type-ahead-city" para carregar as informações.

Em seguida, procede-se à criação de função de ação chamada “registerEmail”. As funções de ação necessitam de algum *trigger* no código de *HTML* para serem lançadas, por exemplo, esta função é executada ao carregar no “Register”.

Esta função vai registar o utilizador na base de dados. Nesta função, são chamadas as validações de primeiro e segundo ecrã, procede-se à criação de um objeto com as informações que o utilizador forneceu, ou seja, informações como email, nome, sexo, data de nascimento, país, distrito, cidade e o *avatar*, sendo só o email, nome e país os campos obrigatórios. Este objeto é enviado com a *API* de “/users/register?token=[token]”, sendo o *token a* juncão de email e *password* no formato encriptado. Na conclusão de registar, é recebido um email de confirmação, que contem um *link* de ativação de conta.

5.3.7 Login

A página de *login*, que é a página principal, serve para autentificar o utilizador.

Route

A entrar na página principal (a página de login), é verificado caso o utilizador já tenha iniciado a sessão antes, ou seja, se fechou o *browser* ou *tab* do *browser* sem fazer *logout*. Para isso usa a função de “restore” do sistema de autentificação, como falado no ponto 5.3.2.

Template

Como o *template* de Login já existe e esta inserido na página principal (“home”), só é necessário adicionar dados novos. Como a Figura 22 mostra, abaixo do botão de “Login with Facebook”, foram adicionados 3 novos elementos, dois campos de *input*, email e *password*, e um botão para *login*.

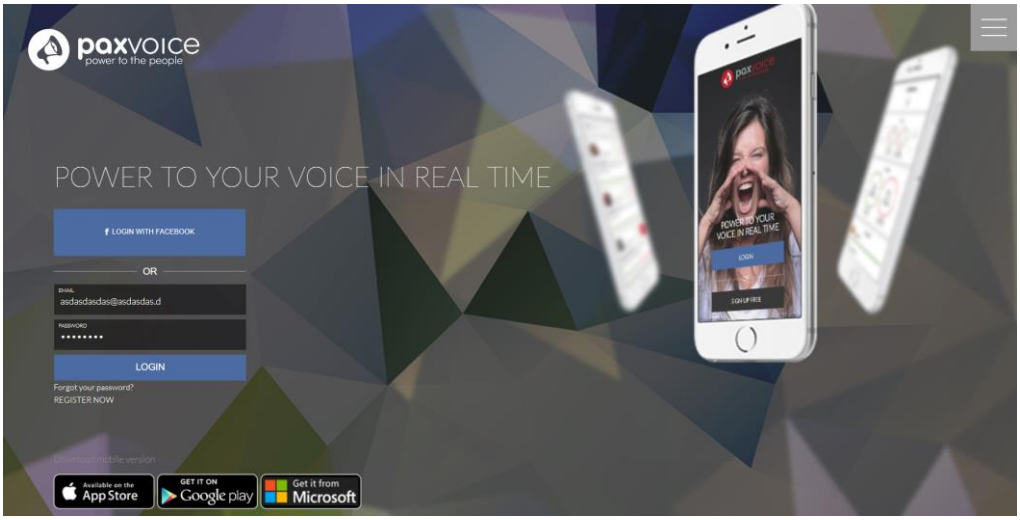


Figura 22 - *Template* de ecrã do Login no PaxVoice

Controller

No *controller* da página principal, que é usado para realizar o *login*, foi adicionada a função para validação dos campos chamada "validateUser", que verifica se o campo email tem formato de um email e se a *password* tem o formato de uma *password*, também validando para os campos não serem vazios. Depois de verificar se a função de validação funciona sem problemas, procedemos à criação de uma função de ação.

A função foi denominada "loginEmail". Função esta que vai ser lançada ao carregar no botão de "LOGIN", situado no *template* de Login. Esta função, ao ser executada, corre a função "validateUser" que valida as informações de utilizador, caso válidas, encripta o email e a *password* do utilizador para criar o *token*. Após o *token* criado, usa a função de "authenticate" do ficheiro "app.js" falado no ponto 5.3.2, esta função autentifica o utilizador caso o email e *password* existam na base de dados, fazendo o *login* do mesmo.

5.3.8 Settings

Na página dos *settings* o utilizador pode verificar e modificar os seus dados pessoais, também podendo mudar a sua *password*.

Model

Como a página de *settings* já existia, só foi necessário modificar o seu *Model*, onde foram adicionadas as informações pessoais de utilizador usando a *API* "/settings?token=[token]".

Template

No *template* de *settings*, antes de aplicar as modificações, ele só tinha um campo para a introdução do email, *checkbox* da opção de receber, ou não, as notificações por email e uma hiperligação para conectar com o *Twitter*.

Adicionou-se então ao *template* existente os campos de upload de imagem, que também permite a sua *preview*, o nome, sexo, a data de nascimento, país, distrito, cidade e os três campos para modificar a *password*: *password* atual, *password* nova e reintrodução de *password* nova. Podemos verificar o resultado final na Figura 23.

No caso dos campos de password, eles só são mostrados no caso de utilizador realizar o *login* a partir de email.

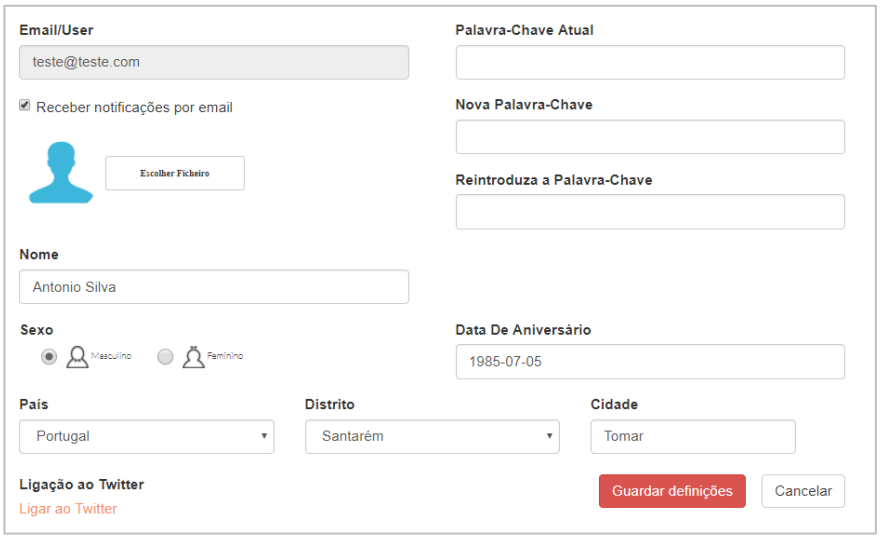


Figura 23 - *Template* dos *Settings* no PaxVoice

Controller

Finalmente, no caso de *controller* dos *Settings*, foram criadas as funções semelhantes às do *controller* de Registar, em específico, as funções que permitem carregar as informações dos países, distritos e cidades.

De seguida foi criada a função “validate”, que valida os campos dos *Settings*, em específico, o nome e país, que não podem ser vazios, o avatar, que tem que ser do tipo imagem, a data de nascimento e as *passwords* (atual, nova e a repetição da nova).

Após criar as funções acima, criou-se a função de ação “submit”, que serve para atualizar as informações pessoais de utilizador. Nesta função, seguimos o processo semelhante ao de registar, criando um objeto, com as informações de utilizador, e enviámos este objeto para a *API* “/settings?token=[token]”, onde o *token* é a juncão de email e da *password* encriptados.

Caso o utilizador tenha feito o *login* com email, pode também modificar a *password*, tendo os campos de *password* (atual, nova e a repetição da nova) preenchidos.

Para modificar a *password*, é comparado o *token* da sessão com o *token* criado na juncão de email e a *password* inserida no campo de *password* atual, caso esta comparação seja válida, ou seja, os *tokens* sejam iguais, a *password* será modificada usando a *API* “/settings/password?token=[token]&newtoken=[newtoken]”, onde o “token” é o *token* da sessão e o “newtoken” é a *token* gerada com o email e a *password* nova.

5.3.9 Recuperação de Password

Nos casos de utilizador esquecer-se de sua *password*, foi necessário criar uma página de recuperação do mesmo.

Na página criada, existe um *input* que é preenchido com email do utilizador. Após o utilizador preencher o campo e submeter o pedido, é enviado um email com um *link* de recuperação da *password*.

Quando o utilizador abre o *link* do email, vai lhe ser apresentada a página para modificar a *password*.

Nesta página, como a Figura 24 mostra, existem dois campos de input, um sendo preenchido com a *password* nova e segundo com a repetição de *password*. Ao confirmar o pedido a *password* é modificada pedindo a utilizador para realizar o *login* usando a nova *password*.

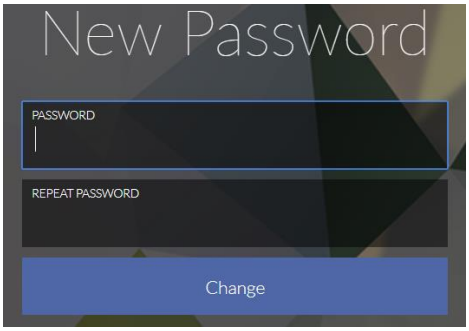


Figura 24 - A página de recuperação da *password*

5.3.10 Ativação de conta e Reenvio de token

Foi criada uma página para ativar a conta, esta página é acedida a partir do *link* do email, recebido após a registar uma conta nova. Ao abrir este *link*, o *controller* da página usa a *API* "/users/validation?token=[token]" onde o *token* é um parâmetro de *link* recebido por email. Caso o processo de ativação decorreu com sucesso, é mostrado uma mensagem a informar do mesmo.

Caso, por alguma razão, o utilizador pode não receba o email de ativação da conta após a registar-se, pode pedir um novo email de ativação, clicando a uma hiperligação chamada "Resend Ativation Email" que se encontra na página de registar.

Nesta página existe um campo de *input*, onde o utilizador adiciona o seu email. Após a confirmar o pedido, é usada a *API* "/users/askValidation?email=[email]", onde o email é o email da *input*, que vai reenviar o email de validação para o email de utilizador.

5.3.11 Múltiplas Línguas

Durante os desenvolvimentos, foi importante criar uma aplicação web que permitisse aos utilizadores não portugueses usar a mesma, para atingir este objetivo, todo o sistema de autentificação foi criado a pensar em suportar mais que um idioma, neste momento sendo suportado o inglês e português. Apesar de suportar só duas línguas no momento, é possível adicionar mais no futuro.

6 Conclusões

Após os nove meses de estágio, faço um balanço positivo desta experiencia. Aprendi muito ao participar nos projetos SouCidadao e PaxVoice. Em específico, vi como funciona o processo do desenvolvimento do ponto de vista da empresa, o processo de divisão de tarefas entre os programadores, a importância do trabalho de equipa, as fases pelas quais o projeto tem que passar para chegar ao consumidor final, e a importância dos testes.

O maior desafio ao longo do estágio foi aprender os fundamentos de tecnologias novas, como *Drupal*, *Ember.js* e *Angular.js*, mas também a falta de experiencia em algumas questões.

Apesar de encontrar estas dificuldades e cometer alguns erros, usando os conhecimentos adquiridos durante o estágio, mestrado, e com a ajuda dos responsáveis no estágio, concluí todos os objetivos propostos e no final do trabalho concluo que apendi muito. Da mesma forma, os conhecimentos obtidos durante a licenciatura de Engenheira Informática foram uma grande ajuda para levar este trabalho a bom porto.

Com este trabalho consegui cumprir os objetivos propostos pela empresa, ao fazer os desenvolvimentos de todos os serviços propostos no SouCidadão e no PaxVoice, conseguindo, no geral, melhorar estas aplicações, juntamente com o resto da equipa. Neste último, foi conseguido responder ao pedido dos utilizadores criando um sistema de autentificação por email.

Em conclusão, este estágio foi uma boa experiencia profissional, onde ganhei muitos conhecimentos e ultrapassei as dificuldades encontradas.

7 Bibliografia

Angular.js. (s.d.). *What Is AngularJS*. Obtido em 1 de novembro de 2017, de https://docs.angularjs.org/guide/introduction

Apache Cordova. (s.d.). *Overview*. Obtido em 2 de novembro de 2017, de https://cordova.apache.org/docs/en/latest/guide/overview/

Apple. (1 de novembro de 2017). Obtido de https://developer.apple.com/xcode/

Beal, V. (s.d.). *Programming Language*. Obtido em 10 de outubro de 2017, de webopedia: https://www.webopedia.com/TERM/P/programming_language.html

Compta. (2017). *1º Semestre de 2017 Informação Intercalar*. Obtido em 24 de outubro de 2017, de http://www.compta.pt/wp-content/uploads/2017/10/Compta1Semestre2017.pdf

Compta. (s.d.a). Obtido em 1 de novembro de 2017, de Compta: http://www.compta.pt/sobre-nos/

Compta. (s.d.b). *Business Solutions*. Obtido em 29 de outubro de 2017, de http://www.compta.pt/en/business-solutions/

Drupal. (6 de dezembro de 2016). *Overview*. Obtido em 2 de novembro de 2017, de https://www.drupal.org/docs/7/understanding-drupal/overview

Ember. (s.d.). *Guides and Tutorials*. Obtido em 1 de novembro de 2017, de https://guides.emberjs.com/v2.16.0/

Ionic. (1 de novembro de 2017). *Overview*. Obtido de http://ionicframework.com/docs/v1/overview/

jQuery. (1 de novembro de 2017). Obtido de https://jquery.com/

PuTTY. (8 de julho de 2017). *PuTTY FAQ*. Obtido em 1 de novembro de 2017, de https://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html

SoapUI. (1 de novembro de 2017). *Overview*. Obtido de https://www.soapui.org/open-source.html

SouCidadão. (s.d.). Obtido em 24 de outubro de 2017, de http://www.soucidadao.pt/

WinSCP. (1 de novembro de 2017). *Introducing WinSCP*. Obtido de https://winscp.net/eng/docs/introduction