UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Sistema de e-Tickets para Serviços de Atendimento

João Filipe Fernandes Fonseca Ribeiro Ramalho

Orientação: Luís Miguel Mendonça Rato

Co-Orientação: Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, Maio de 2018

UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Sistema de e-Tickets para Serviços de Atendimento

João Filipe Fernandes Fonseca Ribeiro Ramalho

Orientação: Luís Miguel Mendonça Rato

Co-Orientação: Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, Maio de 2018

Sumário

Na sociedade actual, os serviços de atendimento ao público são uma presença comum no quotidiano. Sejam serviços públicos ou privados, esperar para ser atendido, sempre foi algo que frequentemente ocorre. Várias foram as soluções encontradas para minimizar e ordenar esta situação, seja através de uma simples fila de espera física, em que literalmente as pessoas esperam por ordem de chegada em fila pela sua vez, ou através de uma fila virtual, formada por uma numeração obtida em função da ordem de chegada. Independentemente da estratégia utilizada, um factor é sempre transversal, a espera presencial pelo atendimento. Este trabalho visa dar resposta às longas filas de espera, que de forma mais ou menos frequentes vão surgindo nos mais variados serviços de atendimento ao público. Estas filas de espera, frequentemente extensas e morosas, têm consequências para as empresas ou instituições que prestam o serviço. Sublinha-se, outrossim, que têm ainda um forte impacto na vida das pessoas que necessitam de recorrer a este tipo de serviço, quer seja pela perda de tempo ou pela impossibilidade de ser atendido. O objectivo, da dissertação que ora se apresenta, é procurar solucionar, ou reduzir de forma considerável o problema aqui descrito. Para o efeito vai ser criado um sistema integrado, com os actuais sistemas de gestão de atendimento, para que a informação e interacção que anteriormente só era possível de forma presencial, passe a ser possível de forma virtual, através de uma aplicação *web* e/ou *mobile*. Este sistema dará em tempo real o tempo de espera estimado por cada serviço e permitirá gerar um *e-ticket* que servirá para dar acesso à sua vez no atendimento. Para tal, é necessário implementar um sistema composto por diversas tecnologias, tais como, *Web Server*, base de dados, API de *webservices* com recurso a *microservices*, de modo a que a interoperabilidade entre sistemas seja assegurada, aplicação com versão *web* e *mobile* que sejam intuitivas, responsivas e que permitam a sua utilização com fracas condições de ligação à internet. Assim, espera-se que após a implementação do sistema, as filas de espera sejam normalizadas (reduzindo os picos), e que cada vez menos exista um período de espera presencial nos serviços de atendimento, dando liberdade às pessoas para continuar com outras actividades, enquanto o momento do seu atendimento não chega.

e-Tickets *System for Customer Services*

Abstract

In today's society, public service departments are a common presence in everyday life. Whether public or private services, waiting to be answered was always something that occurs frequently. Many are the solutions to minimize and organize this situation, either through a single physical queue where literally people expect first-served basis in line for their turn, or via a virtual queue formed by a obtained numbering function of the finishing order. Regardless of the strategy used, one factor is always cross the face waiting for service. This paper aims to address the long queues, which more or less frequent form arise in various public attendance services. These large queues have consequences for companies or institutions that provide the service, but above all have a strong impact on the lives of people who need to resort to this type of service, whether the loss of time or the inability to be attended. The objective is to solve or reduce considerably the problem described herein. For it will be created an integrated system with current service management systems so that information and interaction that was previously only possible in person, go to be possible in virtual form, via a web application and / or mobile. This system will provide real-time estimated wait time for each service and will generate an *e-ticket* which will serve to provide access to the attendance queue. To do this, it is necessary to implement a system composed of several technologies, such as web server, database, webservices API using microservices so that interoperability between systems is ensured, application with web and mobile version that are intuitive , responsive and allow their use with poor internet connection conditions. Thus it is expected that after implementation of the system, queues are normalized (reducing the peaks) and that fewer times anyone needs to wait in attendance services, giving freedom to people to continue with other activities until the exact time of attendace is arrived.

À minha Avó Tina, que tanto queria mas que infelizmente já cá não está para assistir...

Agradecimentos

Agradeço obviamente a todos os que me apoiaram!

Em especial gostaria de agradecer aos meus pais, João Ramalho e Fátima Ramalho, que sempre me apoiaram. À minha avó, Catarina Fonseca, que sempre fez tudo para que eu prosseguisse os estudos. Aos meus sogros, Luís Fouto e Maria Fouto, pelos constantes incentivos. À minha esposa, Raquel Fouto, que sempre me incentivou na conclusão da dissertação e me apoiou dia a dia. Foi um suporte fundamental! Que me seja permitido um agradecimento especial à minha filha Maria Ramalho, que é uma grande motivação na minha vida, e que felizmente já dorme umas noites de seguida. Queria ainda agradecer aos meus orientadores, Professores Vítor Nogueira e Luís Rato, por todo o apoio e disponibilidade no decorrer de todo este processo. Mais uma vez, ao Professor Vítor Nogueira, que nas aulas práticas de programação 1, com o seu entusiasmo e forma de ensinar, desenvolveu em mim um gosto genuíno pela informática em geral e pela programação em particular.

Mais uma vez, obrigado a todos!

Acrónimos

API Application Programming Interface

BASH Bourne-Again SHell

CSS Cascading Style Sheets

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JS JavaScript

JSON JavaScript Object Notation

NoSQL Not Only SQL

PHP Hypertext Preprocessor

PWA Progressive Web Apps

REST REpresentational State Transfer

SOAP Simple Object Access Protocol

SQL Structured Query Language

WBS Work Breakdown Structure

WWW World Wide Web

XML eXtensible Markup Language

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Desde que existem serviços de atendimento ao público, pelo menos em algum momento houve um período de espera. Essa situação ocorre sempre que existe um número de pessoas por atender superior ao número de pessoas para atender. Quando essa situação ocorre são utilizadas duas formas de lidar com o problema, criando uma fila de espera física, em que a pessoa avança na fila até chegar a sua vez, ou criando uma fila de espera virtual, em que se retira uma senha no momento da chegada. Devido à necessidade crescente de optimização de recursos, os agentes disponíveis para atendimento nos mais variados serviços, têm vindo a diminuir. De facto, não é viável ter recursos suficientes para dar resposta a períodos de pico e durante a maior parte do tempo ter recursos em excesso. Essa situação, leva a que invariavelmente surjam filas de espera de grandes dimensões. A origem desta situação é variada, poderá surgir nos últimos dias de uma data limite, lançamento de um novo produto ou serviço, durante a hora de almoço (que normalmente coincide com a de alguns funcionários do serviço)...

1.1 Motivação

Como sabemos, o tempo é um recurso cada vez mais precioso, pelo que tempo perdido em espera é manifestamente um desperdício. Quantas vezes não se perde uma hora de almoço, por vezes em vão, para nos deslocarmos a um destes serviços, ou em última instância somos forçados a utilizar dias de férias para ter a disponibilidade necessária para levar essa tarefa até ao fim. Noutras vezes o atendimento é rápido e nenhum dos problemas supra citados ocorre. Dos vários problemas que manifestamente ocorrem, do ponto de vista do cliente, sublinham-se a imprevisibilidade e tempo de espera para atendimento e consequente obrigação de permanecer no local. Do ponto de vista da entidade que presta o

serviço, reconhecer-se-à haver recursos subaproveitados grande parte do tempo, espaços de atendimento sobredimensionados, possível desmotivação dos funcionários devido, à deterioração do ambiente de trabalho nos períodos de grande afluência. De forma a mitigar as situações anteriormente descritas, surge a necessidade de actualizar o paradigma. Para quê perder tempo na deslocação a um serviço sobrelotado com um tempo estimado de espera de 30 minutos, quando a 5 minutos de distância existe outro serviço idêntico com um tempo estimado de espera de 5 minutos? Para quê perder tempo na deslocação a um serviço, tirar uma senha e depois ter que aguardar horas para ser atendido? Porque não consultar à *priori* qual o serviço com o atendimento mais rápido? Porque não tirar a senha à distância de modo a que o atendimento seja praticamente coincidente com a chegada ao serviço?

1.2 Objectivos

Este projecto tem como objectivo responder a estas necessidades, criando uma plataforma web e móvel de consulta de tempo estimado de atendimento por serviço, e de emissão de *e-tickets* de forma integrada com os sistemas já existentes.

1.3 Abordagem Proposta

O projecto consiste num sistema baseado na *web*, construído de raiz, assente numa estrutura que permita um conjunto de servidores com uma arquitectura altamente escalável, de modo a estar preparado para os desafios de uma utilização intensiva. A *interface* deste sistema é um dos pontos-chave, pois deverá ser extremamente intuitiva, atractiva e multiplataforma, estando acessível via browser e através de dispositivos móveis (aplicação para iOS e Android).

1.4 Principais contribuições

As principais contribuições desta dissertação são:

- Sistema agregador, que permite a integração de todas as plataformas similares de sistemas de atendimento;
- Inovação, permitindo integrar na plataforma sem grande esforço, qualquer serviço existente, sem que para isso seja necessário implementar o seu próprio sistema;
- Implementação de versão web e para dispositivos móveis, que permita a sua utilização em diversos cenários, de forma fácil e intuitiva, sem constrangimentos causados por uma fraca ligação à internet;

- Possibilidade de utilização, por exemplo, nos Serviços Académicos da Universidade de Évora, com o potencial de melhorar bastante as condições de trabalho dos funcionários deste serviço, e sobretudo reduzindo o tempo de espera dos alunos em alturas de pico de atendimento;

1.5 Organização

Esta dissertação está organizada do seguinte modo:

- **Capítulo 1** introdução ao trabalho realizado, as suas motivações, os objectivos que se pretendem alcançar, a forma como se procura atingi-los, e as principais contribuições deste projecto.
- **Capítulo 2** enquadramento da problemática abordada nesta dissertação, são descritas várias soluções e sistemas equivalentes. São também caracterizadas diversas alternativas nas várias tecnologias necessárias para a implementação deste projecto.
- **Capítulo 3** apresentação da proposta para a implementação, listados e descritos os objectivos que se desejam alcançar, assim como a metodologia de trabalho no decorrer deste projecto. São também apresentadas as opções propostas para utilização nas diversas tecnologias necessárias.
- **Capítulo 4** descrição do processo de implementação e ilustrados os vários componentes do sistema proposto.
- **Capítulo 5** avaliação dos resultados alcançados neste projecto, tendo em consideração os objectivos propostos. São também referidos os trabalhos futuros, considerados relevantes na evolução deste projecto.

Capítulo 2

Estado da Arte

Neste capítulo, faz-se um enquadramento da problemática abordada nesta dissertação. São descritas várias soluções e sistemas equivalentes. Serão ainda caracterizadas diversas alternativas, nas várias tecnologias necessárias para a implementação deste projecto.

2.1 Enquadramento

Existem neste momento diversas abordagens de modo a minimizar os problemas anteriormente descritos. Todavia, todos podem ser categorizadas em dois grandes grupos: filas físicas e virtuais. O primeiro grupo, corresponde a uma simples fila em que as pessoas se vão ordenando por ordem de chegada, e o atendimento é feito por essa ordem. O segundo, representa uma fila virtual, em que a cada posição é atribuído um número incremental, que tal como no grupo anterior obedece à ordem de atendimento. Até há pouco tempo, as únicas evoluções visavam melhorias tecnológicas na atribuição de senhas, na interface de visualização do estado actual no atendimento e sistema de chamada. Estas alterações, apesar de melhorarem o serviço, como se reconhecerá, não permitiram dar o salto qualitativo que se impunha, para as necessidades actuais de alguns serviços de atendimento. Recentemente começaram a surgir novas soluções, rompendo com o paradigma existente, em que a posição na fila de atendimento pode ser obtida remotamente, através de um sistema de marcação[Portugal, 2013], sistema de envio de SMS com informação sobre o tempo de espera para um determinado serviço[de Notícias, 2006]. Porém, sendo ainda implementados pontualmente, mostram-se pouco flexíveis e consomem recursos extra. Neste momento, coincidência ou não, começam a surgir novos sistemas que convergem com o trabalho descrito neste texto. O Mapa do Cidadão[AMA, 2015] foi o primeiro. Este sistema permite, consultar em alguns serviços de atendimento, em tempo real o estado das filas de

espera e em alguns casos, a possibilidade de obtenção de uma senha *on-line*. Este serviço, disponibiliza um portal *web*[WWW, 2014] e para dispositivos móveis[Computing, 2014] nos sistemas operativos iOS, Android e Windows Phone. O Mapa do cidadão é baseado no SIGA (Sistema Integrado de Gestão de Atendimento) [SIGA, 2017], que permite obter informação em tempo real sobre o estado dos vários serviços que utilizavam este sistema. O SIGA foi inicialmente implementado nas lojas do cidadão, e aos poucos tem vindo a ser replicado nos restantes serviços de atendimento do sector público. Actualmente já possui aplicação própria[sigaApp, 2017], para *smartphones* com sistemas operativos iOS e Android. É direccionada exclusivamente para a gestão de filas de espera dos serviços que utilizam este sistema, permitindo tirar uma senha *on-line*, acompanhar em tempo real o atendimento e a respectiva estimativa do tempo médio de espera. O sistema Mapa do cidadão, é uma versão mais abrangente do sistema SIGA, pois para além das funcionalidades e informações deste sistema, disponibiliza ainda a caracterização e informação dos restantes serviços existentes do sector público. Recentemente, a Câmara Municipal de Lisboa[CML, 2017], disponibilizou a aplicação móvel Loja Lisboa[CML, 2017] para Android e iOS, que permite o mesmo sistema para as lojas de atendimento municipal. Os problemas relacionados com a gestão de atendimento não são exclusivos do sector público, também no sector privado existe a percepção da necessidade de mudança do paradigma, optimizando cada vez mais os recursos, proporcionar um melhor serviço, e com isso fidelizar cada vez mais os clientes. Exemplo disso, é o lançamento da aplicação móvel tira-vez[SONAE, 2017] que permite emitir senhas e acompanhar os vários serviços de atendimento das lojas Continente[Continente, 2017]. O grupo Sonae[SONAE, 2017], no ano de lançamento da aplicação registou que foram feitos 32000 *downloads*[SONAE, 2017], números que ajudam a confirmar o sucesso da aposta. Tendo em consideração o surgimento destes sistemas, não será de estranhar cada vez mais o seu aparecimento. No sector público, através do seu alargamento a todos os serviços. No sector privado, a concorrência fará com que os restantes retalhistas disponibilizem sistemas equivalentes e possivelmente um dia, acabará por fazer parte de todos os serviços de atendimento. Assistimos actualmente a alguma dispersão nestes sistemas, tendencialmente cada vez maior. Para mitigar essa dispersão, torna-se importante a existência de um sistema, que para além de funcionar de forma autónoma, consiga também ser agregador. Daí resulta que permita integrar todos os sistemas existentes, para que assim o utilizador, não seja obrigado a instalar e/ou consultar uma aplicação por cada serviço de atendimento. Infelizmente esse sistema ainda não existe. Para implementar um sistema deste tipo, é necessário recorrer a algumas tecnologias.

2.2 Tecnologias

De seguida estão identificadas e comparadas as principais alternativas para cada uma das tecnologias utilizadas.

2.2.1 Servidor Web

Um servidor *web*[Servidor Web, 2017] é o *software* responsável pela disponibilização de conteúdos na *web*[WWW, 2014], permitindo assim que, neste caso em particular, utilizando o protocolo HTTP[Hypertext Transfer Protocol, 2018] a aplicação esteja disponível tanto para o utilizador final, como para aplicações externas. Será por isso parte fundamental neste projecto. Existem vários servidores *web* disponíveis actualmente, de todos, os mais utilizados são o APACHE[Apache, 2018], Microsoft IIS[Microsoft IIS, 2018] e NGINX[NGINX, 2018]. Temos ainda o Node.js[Node.js, 2018], que apesar de não ser tão usado, globalmente está em crescimento e é bastante utilizado sobretudo em sites cujo tráfego é mais elevado[w3techs, 2018].

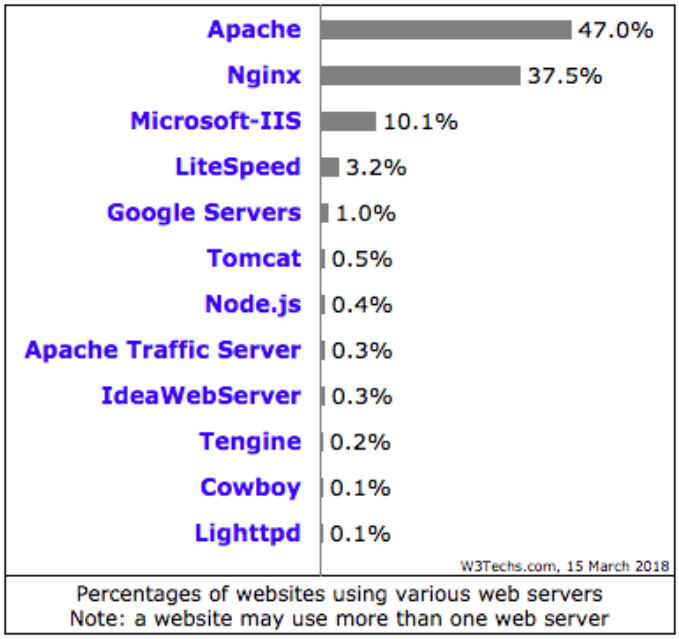

Figura 2.1: Representação da utilização dos vários servidores[w3techs, 2018]

Para este tipo de projecto era importante a utilização de tecnologias *open-source*[Open-source, 2018], pelo que a utilização de Microsoft IIS está fora de questão. A estabilidade e segurança é também um factor importante. Assim o Apache e o NGINX estão mais bem posicionados. Ambos muito idênticos, embora o NGINX tenha melhor desempenho no carregamento de conteúdo estático. O Apache tem a vantagem de ser o servidor *web* mais utilizado, com uma grande comunidade, muitas bibliotecas, sendo uma aposta segura. Em sentido oposto, o Node.JS é mais recente, tem tido um grande crescimento sobretudo em sites cuja utilização é mais intensiva, embora não seja nativamente um servidor *web*, pode funcionar como tal. O elevado desempenho tem cativado cada vez mais utilizadores.

Existem algumas abordagens interessantes, como por exemplo a utilização do NGINX como balanceador de carga[Load Balancing, 2018] do Apache e ainda a utilização de Node.JS para responder a pedidos do Apache, extraindo desta forma o melhor de cada.

Web Server	Nginx	Apache	Node.JS
Supported OS	Linux, Unix, MacOS, Windows-partial	Linux, Unix, Windows, MacOS	Linux, Unix, Windows, MacOS
User support & Fixes	User community only	Corporate & User community	User community only
Cost & Development	Free, Open source	Free, Open source	Free, Open source
Security	Very good	Excellent	Bad
Features & Documentation	Good	Excellent	OK
Performance	Excellent	Good	Excellent

Tabela 2.1: Comparação entre os servidores Apache, NGINX e Node.JS[UpGuard, 2017]

2.2.2 Linguagem de Programação

Como os conteúdos disponibilizados neste serviço são dinâmicos, é necessário recorrer a uma ou várias linguagens de programação de modo a permitir a sua disponibilização, assim como a implementação da aplicação no servidor. PHP[PHP, 2014], C# via ASP.NET[C Sharp, 2018] e Java[Java, 2018], são algumas das mais utilizadas linguagens de programação *server-side*, JavaScript[JavaScript, 2018] em *client-side*.

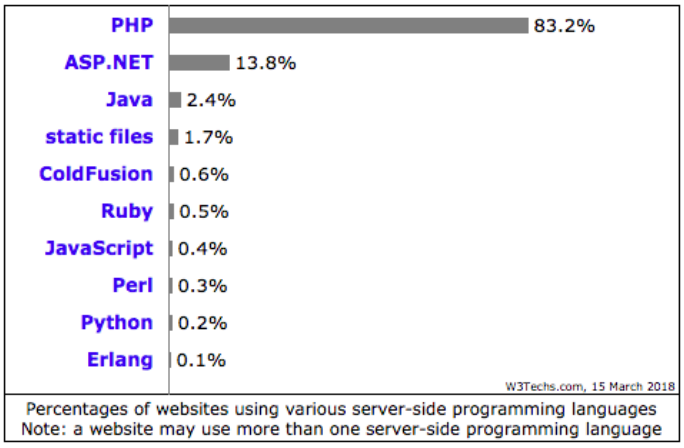

Figura 2.2: Representação da utilização das várias linguagens de Programação *server-side*[w3techs, 2018]

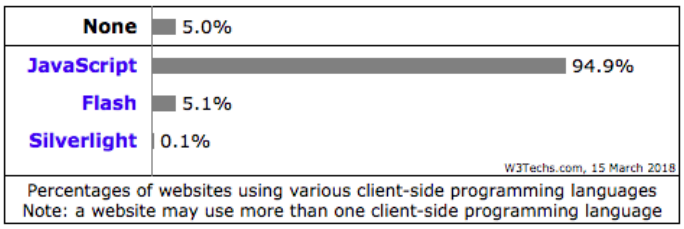

Figura 2.3: Representação da utilização das várias linguagens de Programação Client-Side[w3techs, 2018]

PHP é uma linguagem *open-source*, com uma grande comunidade envolvida, fácil de utilizar, funciona muito bem com APACHE e MySQL, que apesar de não ser nativo, pode funcionar orientada a objectos. Existe uma grande variedade de bibliotecas disponíveis,

frameworks de desenvolvimento, permitindo uma grande flexibilidade na sua utilização aliando uma ágil implementação. Quanto a *client-side*, não existe grande alternativa, JavaScript é a norma, podendo depois variar na utilização de uma ou mais bibliotecas.

2.2.3 Base de Dados

O armazenamento dos dados é fundamental para o funcionamento deste serviço. Na caracterização dos vários sistemas disponíveis, passando pelo armazenamento de informação dos utilizadores, até ao momento da gestão dos vários *e-tickets* emitidos, a informação necessita ser armazenada e relacionada. Assim, uma base de dados relacional ou parcialmente relacional é necessária. Oracle[Oracle, 2018], MySQL[MySQL, 2014], Microsoft SQL Server[Microsoft SQL Server, 2018], são as mais utilizadas.

341 systems in ranking, March 2018

Rank Mar 2018	Rank Feb 2018	Rank Mar 2017	DBMS	Database Model	Score Mar 2018	Score Feb 2018	Score Mar 2017
1.	1.	1.	Oracle	Relational DBMS	1289.61	-13.67	-109.89
2.	2.	2.	MySQL	Relational DBMS	1228.87	-23.60	-147.21
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1104.79	-17.25	-102.70
4.	4.	4.	PostgreSQL	Relational DBMS	399.35	+10.97	+41.71
5.	5.	5.	MongoDB	Document store	340.52	+4.10	+13.59
6.	6.	6.	DB2	Relational DBMS	186.66	-3.31	+1.75
7.	7.	7.	Microsoft Access	Relational DBMS	131.95	+1.88	-0.99
8.	8.	↑ 10.	Redis	Key-value store	131.22	+4.21	+18.22
9.	9.	↑ 11.	Elasticsearch	Search engine	128.54	+3.23	+22.32
10.	10.	↓ 8.	Cassandra	Wide column store	123.49	+0.71	-5.70

Figura 2.4: Representação da utilização das várias Bases de Dados[DB-Engines, 2018]

Apesar das três serem relacionais, apenas a MySQL é *open-source*. É também a que melhor integra a utilização do Apache e funciona muito bem com PHP. A utilização de uma base de dados não relacional pode ser interessante neste projecto, pela alta escalabilidade e desempenho, em simbiose com uma relacional.

2.2.4 Serviços Web

De modo a assegurar a interoperabilidade aplicacional é necessário ter uma boa biblioteca de serviços *web*, que permita uma fácil integração assim como obter uma camada de abstração, dando transparência na sua utilização, sobretudo num ecossistema potencialmente tão heterogéneo como este. Este sistema tem que estar preparado para interagir com uma grande panóplia de sistemas de gestão de atendimento, e ainda aplicações móveis. As metodologias mais utilizadas actualmente são o SOAP (Simple Object Access Protocol)[SOAP, 2018] e o REST (Representational State Transfer)[REST, 2018]. O SOAP é um protocolo, sendo também por isso mais formal, assenta na comunicação através de mensagens no formato XML (Extensible Markup Language) [XML, 2018] e

normalmente faz-se acompanhar de um WSDL (Web Services Description Language) [WSDL, 2018] que descreve os serviços disponíveis e especifica a sua utilização. REST[REST, 2018] é um conjunto de restrições e princípios sobre a arquitectura WWW[WWW, 2014], mais ligeiro e versátil, que normalmente utiliza JSON (JavaScript Object Notation)[JSON, 2014] embora, possam ser utilizados outros formatos, como por exemplo XML ou apenas texto. Uma das grandes vantagens, para além da sua flexibilidade e a elevada eficiência na troca de mensagens entre sistemas, consumindo pouca largura de banda, é a facilidade com que é feito o *parse* das mensagens, já que muitas linguagens possuem métodos nativos para esse fim. Como desvantagem, não sendo um protocolo, e por isso menos rígido na sua implementação, existe o risco de, caso a implementação não seja bem feita, haver problemas de segurança ou de integridade dos dados. Para mitigar esses problemas, existem ferramentas para ajudar na implementação e na documentação dos *webservices* disponíveis. A ferramenta Swagger [Swagger, 2018] é uma das mais utilizadas, permitindo a criação de uma API (Application Programming Interface)[API, 2006] mais segura, documentada e eficiente. Recentemente tem-se verificado cada vez mais a utilização de GraphQL[GraphQL, 2018] que, não sendo também um protocolo, é uma linguagem que permite pedidos filtrados, obtendo assim apenas a informação desejada em cada pedido. Sendo mais estruturada e formal que REST, é normalmente mais seguro, estável e potencialmente proporciona uma melhor integridade dos dados, conjugado com uma superior eficiência, resultante da troca de mensagens sem informação desnecessária em cada pedido.

2.2.5 Interface do Utilizador

Num sistema deste tipo, a usabilidade é um dos pontos chave para o seu sucesso. É necessário ter um design apelativo, simples, intuitivo e responsivo[Responsive, 2018]. Assim, a utilização de uma *framework* para HTML, JavaScript e CSS[CSS, 2018] na sua implementação torna-se fundamental. Dessa forma é possível agilizar o processo, não comprometendo a consistência aplicacional, atingindo elevados níveis de usabilidade, e ainda o suporte da comunidade envolvida no seu desenvolvimento. Actualmente a Bootstrap[Bootstrap, 2018] é a mais mais utilizada[FEF, 2018], tendo-se tornado a referência nesta matéria e por esse motivo é a que tem mais documentação disponível, *templates*, extensões e a maior comunidade. Existem muitas alternativas, de entre as mais utilizadas[FEF2, 2018] estão a Foundation[Foundation, 2018] que alega ser a mais avançada das *frameworks*, conta com apoio da ZURB[ZURB, 2018] que assegura a sua evolução e suporte. Semantic UI[Semantic UI, 2018] que tem como grande mais valia a utilização de sintaxe baseada na linguagem natural, permitindo uma mais fácil interpretação da implementação. Pure[Pure, 2018] que tem como trunfo uma utilização bastante optimizada, reduzida e por isso mais ligeira, desenvolvida a pensar sobretudo nos dispositivos móveis.

2.2.6 Aplicação Móvel

A existência de uma aplicação móvel, ainda que a construção de uma aplicação web responsiva[Responsive, 2018] venha reduzir essa necessidade, é de extrema importância, sobretudo pelo funcionamento *offline*, e a disponibilização de notificações no dispositivo. Existem várias abordagens possíveis, de entre elas, podemos recorrer à implementação de :

- Aplicações nativas puras, que obrigaria ao desenvolvimento de uma aplicação para Android e outra para iOS, mas teriam um melhor desempenho e utilização dos recursos do dispositivo.
- Aplicações nativas com disponibilização de conteúdos através de *webviews*[CSS, 2018], que basicamente consiste na colocação de páginas *web* no interior da aplicação, permitido manter um melhor desempenho global e utilização dos recursos do dispositivo. A mais valia desta solução é reutilização de código através das *webviews*.
- Apache Cordova[Apache Cordova, 2018], que é uma framework que permite transformar conteúdo *web* (HTML+CSS+Javascript) e exportar para uma aplicação móvel. Esta solução é como criar uma aplicação nativa, em que o único conteúdo é uma *webview*. Tem como vantagem a reutilização de código, sobretudo se a aplicação *web* tiver sido implementada com design *responsive*. Tem como desvantagem a dificuldade na utilização *offline*, nos recursos do dispositivo e algumas dificuldades no carregamento de conteúdos (quando existe navegação entre páginas na aplicação e o código Javascript não é totalmente partilhado).
- React Native[React Native, 2018], também é uma *framework* que permite exportar aplicações móveis, mas tem um funcionamento totalmente diferente. A implementação é feita em React JS. Basicamente funciona como um compilador que transforma o código numa aplicação quase nativa. Permite por isso, as vantagens das aplicações nativas, com a vantagem adicional da reutilização de código.
- Progressive Web Apps (PWA)[PWA, 2018], tem uma abordagem interessante. É uma evolução da *web app*, que ganha funcionalidades adicionais, como o funcionamento *offline*, assenta na utilização de *service workers*, que tratam da gestão dos pedidos e respectivos conteúdos, permite também o acesso a recursos do dispositivo e cria um ícone de aplicação. Tem como vantagem, a utilização do mesmo código tanto, na versão *web* como móvel e melhora a utilização na versão *web* (devido a uma utilização eficiente da *cache*). Como desvantagem, tem o facto de não ser uma aplicação nativa. Daí decorrem as já referidas perdas no desempenho e dificuldades na utilização de recursos do dispositivo, sobretudo em iOS.

Capítulo 3

Proposta

Neste capítulo faz-se uma apresentação da proposta para a implementação, listados e descritos os objectivos que se desejam alcançar, assim como a metodologia de trabalho no decorrer deste projecto. São também apresentadas as opções propostas para utilização nas diversas tecnologias necessárias.

3.1 Apresentação

Este projecto tem como objectivo responder à problemática anteriormente descrita, através da criação de um sistema altamente escalável de modo a estar preparado para os desafios de uma utilização intensiva com interfaces[Interface, 2014] *web*[WWW, 2014] e móvel[Computing, 2014], de consulta de tempo estimado de atendimento por serviço e de emissão de *e-tickets* de forma integrada com os sistemas já existentes.

3.1.1 Objectivos

De modo a dar sequência ao plano de trabalho é necessário antes especificar os objectivos:

- Criação de sistema de *e-tickets* para serviços de atendimento
- Ambiente *Web*[WWW, 2014] e Móvel[Computing, 2014]
- Complemento aos actuais sistemas de gestão de atendimento
- Integrar sistemas idênticos, funcionando como agregador

- Dados disponíveis em tempo real[Real-time, 2014]
- Multiplataforma[Multiplataforma, 2014]
- Apelativa e Intuitiva
- Tempo de espera estimado por serviço de atendimento
- Gerar *e-tickets* únicos
- Segurança, autenticação e tipos de acesso
- API de Serviços *Web*[API, 2006] para integração com os vários sistemas existentes e para interacção dos serviços *web*[WWW, 2014] e aplicações móveis[Computing, 2014]
- Requisitos :
  - Utilização em *smartphones*
  - Acessível através de todos os *browsers*[Navegador, 2014]
  - Plataforma assente num sistema altamente escalável e de grande acessibilidade
  - Permitir concorrência nos pedidos
  - Ligação à internet para actualização de dados e pedidos de *e-tickets* e posteriormente possibilidade de utilização *offline*[Offline, 2006] nas aplicações móveis[Computing, 2014]

3.1.2 Plano de Trabalho

De acordo com os objectivos acima descritos, passamos à descrição do plano de trabalho:

- Concepção – Período dedicado a investigação, levantamento de todos os dados, especificação detalhada dos trabalhos e pormenorização dos objectivos a atingir
  - Especificação
    - Descrição dos trabalhos a realizar
  - Elaboração de documento
    - Utilização da informação reunida de forma a iniciar a documentação do trabalho
- Desenvolvimento – Período de trabalho prático, registo de evolução e actualização da documentação
  - Implementação do servidor, serviços *web* e respectiva API[API, 2006]
    - Parte fundamental do sistema. Permitir a integração com os vários sistemas de gestão de atendimento, com as interfaces *web* e com dispositivos móveis. Desenvolvido em PHP[PHP, 2014] e utilização de base de dados MySQL[MySQL, 2014]. A comunicação será feita utilizando objectos JSON[JSON, 2014], e a arquitectura do servidor de modo a permitir escalabilidade e redundância.

        * Prototipagem visual e funcional dos diversos interfaces[Interface, 2014]
            · Criação de *layouts*[Layout, 2014] transversais a todos os interfaces[Interface, 2014], sempre com o objectivo de permitir uma experiência de utilização fácil e intuitiva[Friendly, 2014]
        * Implementação de interface[Interface, 2014] *Web*[WWW, 2014]
            · Criação de site segundo a abordagem *Responsive web design*[Design, 2014], com funcionalidade de consulta de tempo estimado de espera por serviço, e gerar *e-tickets*.
        * Implementação de aplicação Android[Android, 2014]
            · Criação de aplicação em Android[Android, 2014], que para além das funcionalidade oferecidas no interface[Interface, 2014] web[WWW, 2014], permite guardar *e-tickets* para apresentar no serviço de atendimento e utilização *offline*[Offline, 2006]
    – Adaptação dos sistemas actuais para permitir a integração com o novo sistema
        * Especificação – descrição do modo de integração
        * Elaboração de documento – documentação do sistema e respectiva API[API, 2006] para permitir a integração por parte de qualquer sistema de gestão de atendimento

- Testes Finais
    – Integração de sistemas – testar a integração entre os vários sistemas
    – Usabilidade – utilizando um grupo de teste, obter resultados da utilização e fazer ajustes caso necessário, tendo em vista uma melhor experiência de utilização do serviço
    – Qualidade dos dados – Comparar valores reais com valores estimados e proceder a afinações caso necessário
    – Carga – realizar testes de carga, para avaliar resposta em situações de sobrecarga do sistema. Em função dos resultados, delinear plano correctivo e/ou de contingência.

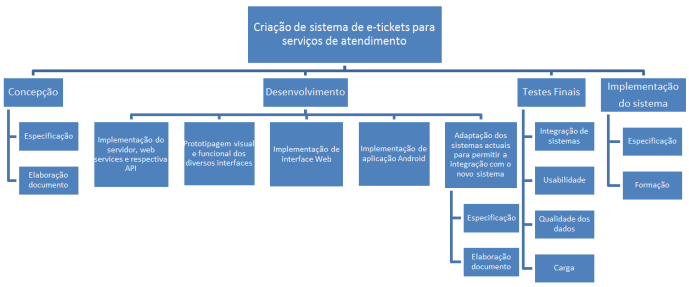

Figura 3.1: Representação do plano de trabalhos recorrendo a um WBS (Work breakdown structure)[WBS, 2014]

Assim temos esta planificação de forma esquematizada na figura 3.1.

3.2 Tecnologias

Foram apresentadas várias hipóteses nas diversas tecnologias necessárias para o desenvolvimento deste projecto. De seguida será feita a apresentação de cada uma das escolhas.

3.2.1 Servidor Web – Apache

O Apache é um servidor HTTP *open-source*. É o mais utilizado nas últimas duas décadas. Conta com uma grande comunidade, que ajudou ao longo dos anos no seu desenvolvimento. Possui uma grande variedade de módulos, bibliotecas e documentação disponível. Não é o mais rápido nem eficiente. Porém, permite elevados níveis de segurança, facilidade e flexibilidade na sua utilização, e ao contar com uma grande comunidade envolvida, temos a garantia de suporte e atualização. Pode ser utilizado em paralelo com outros, permitindo desta forma, a utilização das vantagens de cada um. Nesta primeira fase de desenvolvimento, propõe-se a sua utilização para uma implementação mais célere, ao mesmo tempo mais completa e segura. Esta decisão, tal como já referido, não inviabiliza a utilização futura de outros servidores, como por exemplo o NGINX para funcionar como balanceador de carga e disponibilização de conteúdo estático, melhorando posteriormente o desempenho aplicacional.

3.2.2 Base de Dados – MySQL

MySQL Server é uma das mais utilizadas. Conta com muitos anos de desenvolvimento e uma grande comunidade a apoiá-la. É uma aposta segura para esta fase do projecto, funcionando muito bem com Apache. Posteriormente, é possível adoptar a implementação, sem grandes constrangimentos de MySQL Cluster, tanto em modo paralelo ou em substituição. Esta solução permite um *cluster* de bases de dados, *sharding* e funcionamento autónomo com API NoSQL, como por exemplo Node.JS. Desta forma, quando assim se justificar, é possível implementar um sistema altamente escalável e de alta disponibilidade.

3.2.3 Linguagem de Programação – PHP

PHP é uma linguagem interpretada, *server-side*. A utilização de PHP em conjunto com Apache e MySQL é uma combinação bastante utilizada. O seu funcionamento é bastante estável, conta com muita documentação disponível, *frameworks* e é bastante fácil de utilizar. Com o lançamento da versão 7, houve uma grande evolução (melhorou muito o desempenho e as funcionalidades disponíveis).

3.2.4 API de Serviços Web – Swagger

De modo a criar uma camada de abstração aplicacional, será utilizada uma API que documente todos os serviços disponíveis, funcionando de forma autónoma, como microserviços[Microservices, 2018], permitindo interoperabilidade entre os vários sistemas, sejam eles o *front-end* da aplicação *web*, aplicações móveis, sistema de administração ou de gestão de atendimento. Para a criação dessa API, escolheu-se a ferramenta SwaggerUI, que permite a sua criação de forma fácil, estruturada e segura.

3.2.5 Interface do Utilizador – Bootstrap

O Bootstrap é uma *framework* de desenvolvimento de *front-end*. Permite agilizar o processo de desenvolvimento, uma implementação de forma responsiva, atrativa e de fácil utilização. Dispõe de muitos *templates* prontos a utilizar e adaptar, tal como, muita documentação e actualizações constantes, de forma a compatibilizar com as actualizações nos diversos *browsers*. É por isso indicada para este projecto, pois permite de forma ágil e eficiente atingir os objectivos propostos.

3.2.6 Aplicação Móvel – PWA e React Native

O primeiro passo será a implementação de uma aplicação *web* responsiva. Desta forma, é possível ter uma versão adaptada às diversas resoluções dos vários dispositivos móveis. Esta estratégia, permite com um único desenvolvimento ter uma versão compatível com todos os pontos de acesso. Ainda assim, esta abordagem tem limitações, nomeadamente a utilização *offline*, a necessidade de acesso via *browser*, um sistema de *cache* pouco optimizado, obrigando por isso ao carregamento de muitos conteúdos estáticos, sem necessidade. Para mitigar estas limitações, propõe-se a utilização de PWAs, permitindo desta forma melhorar o desempenho da aplicação *web*, através de um sistema de *cache* optimizado. Permite também o seu funcionamento em modo *offline*. Assim, temos uma melhor experiência de utilização *web*, e ao mesmo tempo uma aplicação móvel. Ainda assim, uma aplicação nativa é recomendada, para um melhor desempenho, utilização *offline* e utilização dos recursos do dispositivo. Esse desenvolvimento não inviabiliza a implementação anterior, pois existe uma melhoria na aplicação *web*. Para a implementação da aplicação móvel, recomenda-se a utilização de React Native, que permite a exportação para Android e iOS, com o mesmo código em React JS. Permite também, a utilização de código nativo, caso seja necessário. Desta forma, num único projecto temos as duas aplicações.

Capítulo 4

Implementação

Neste capítulo, descrevemos o processo de implementação dos vários componentes do sistema proposto. Os métodos de instalação e configuração, aplicam-se no sistema operativo Ubuntu Server 16.04.4 LTS. De seguida, na figura 4.1, temos representado um esquema ilustrativo da arquitectura do sistema implementado.

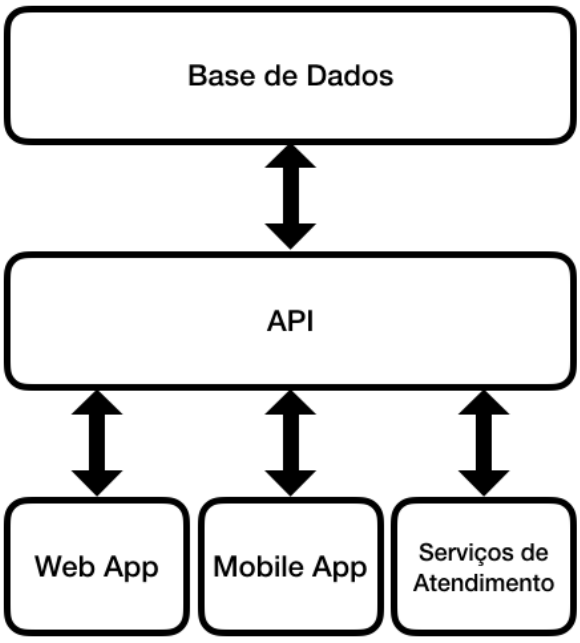

Figura 4.1: Arquitectura do sistema

4.1 Servidor Web – Apache

A instalação do servidor web Apache é bastante simples. Basta executar na *bash*[Bash, 2018] os seguintes comandos:

- "sudo apt-get update"
  - Actualiza a lista de pacotes
- "sudo apt-get install apache2"
  - Instala o Apache
- "sudo ufw allow http"
  - Abre a porta 80 para pedidos ao servidor

Para efeitos de implementação vamos utilizar as definições por defeito. A pasta do servidor por defeito é "/var/www/html". Caso não seja alterada, será aí que se deve implementar o sistema.

4.2 Base de Dados – MySQL

Tal como a instalação do Apache, a instalação da base de dados MySQL também é bastante simples, bastando para isso executar o seguinte comando :

- "sudo apt-get install mysql-server"

E seguir os passos. Durante o processo será definida a palavra passe de *root*. No final da instalação, estão reunidas as condições para começar a desenhar e implementar a base de dados. A base de dados criada é bastante simples, representando nas suas tabelas cada um dos objectos necessários, para a implementação deste sistema:

- "entity": representa uma organização
- "service": representa uma divisão de uma organização
- "servicedesk": representa um balcão de atendimento de uma divisão
- "users": representa um utilizador
- "eticket": representa uma senha para um utilizador num balcão de atendimento
- "countries": tabela auxiliar que contém todos os países disponíveis.

Na figura 4.2, temos a representação do modelo relacional da base de dados criada.

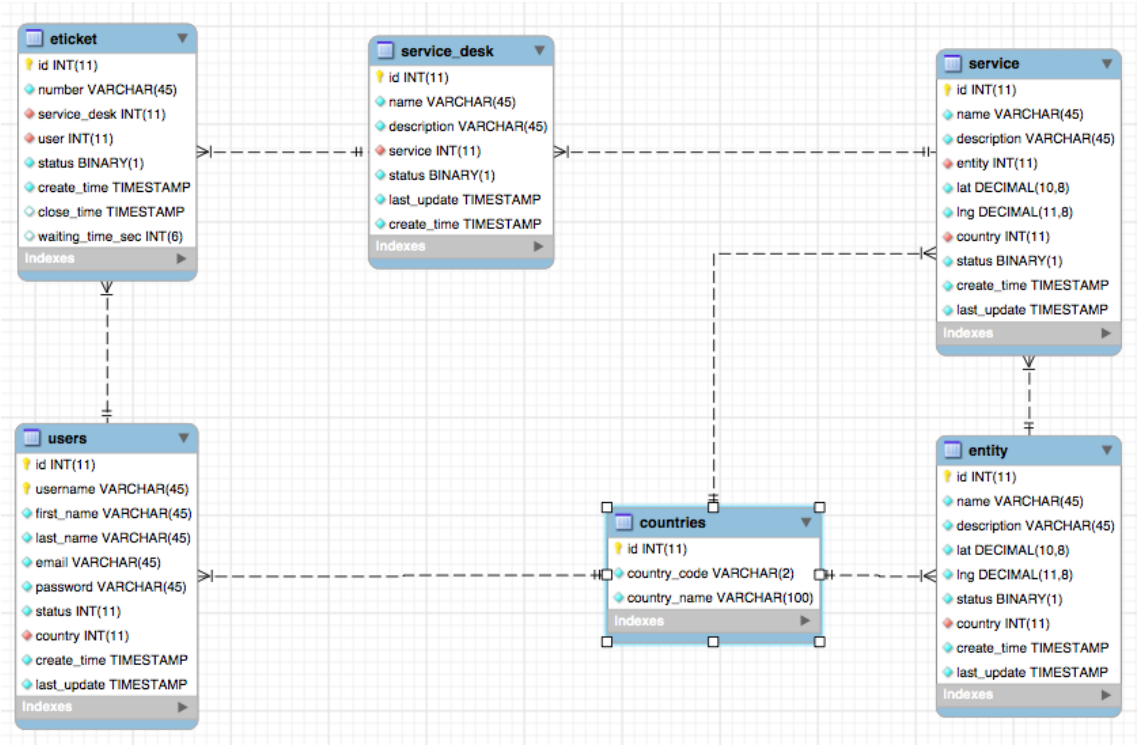

Figura 4.2: Representação do modelo relacional da base de dados

4.3 API de Serviços Web

A API tem um papel fundamental na estrutura deste projecto. É através dela, pelos microserviços existentes, que é possível criar uma camada de abstração, que permitirá garantir a interoperabilidade entre os vários sistemas. Assim, a aplicação *web*, móvel, serviços de atendimento e outros serviços idênticos podem ser integrados facilmente, recorrendo a esta interface. Os vários serviços criados para esta API, podem comunicar entre si ou de forma autónoma para a disponibilização de um serviço. A ligação à base de dados é assegurada por esta, e permite que no futuro a arquitectura da base de dados seja totalmente redesenhada, sem que com isso se reflita qualquer constrangimento no funcionamento aplicacional, dando por isso transparência na sua utilização. Podemos por exemplo, ter serviços que utilizam uma base de dados relacional, outros que utilizam uma não relacional e até que utilizem várias, como podemos ver na figura 4.3.

Para a criação da interface HTML foi utilizado o editor *online* Swagger[Swagger Editor, 2018], assim a tarefa de a criar, tal como a sua documentação, ficou mais expedita. Desta forma os esforços puderam ser concentrados na sua implementação, bastando depois descrever os vários serviços, caracterizando os diferentes níveis de acesso no editor, de acordo com a notação utilizada por esta ferramenta.

Na figura 4.4, temos um *screenshot* da interface da API, com os vários serviços disponíveis.

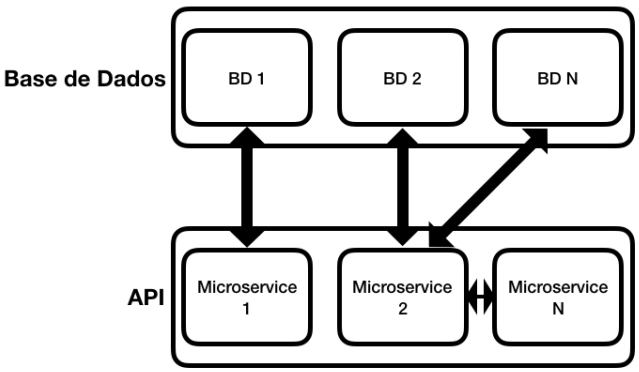

Figura 4.3: Exemplo de várias formas de interacção entre API constituída por micro-serviços e bases de dados

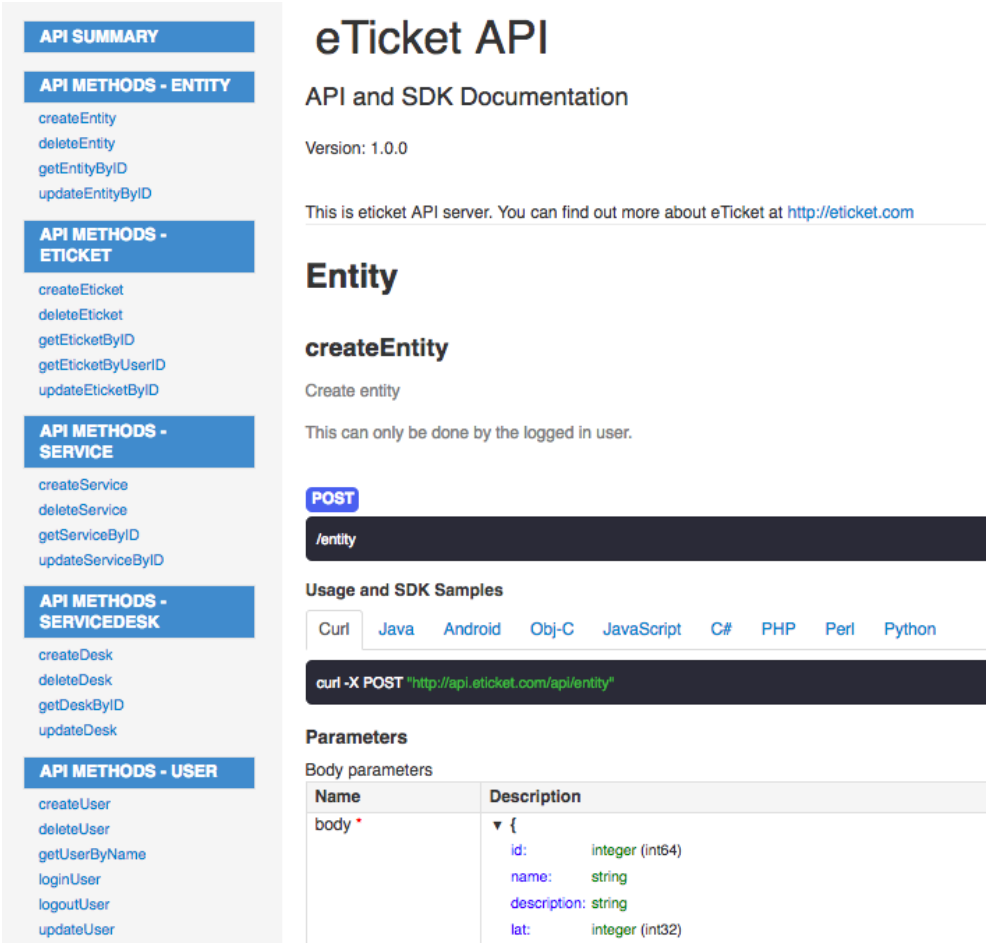

Figura 4.4: *Screenshot* da interface da API

4.4 Aplicação Web

De modo a agilizar o desenvolvimento da aplicação *web*, foram utilizados dois *templates* Bootstrat, Avilon[Avilon, 2018] para a *landing page* da aplicação e NiceAdmin[NiceAdmin, 2018] para a parte aplicacional. Desta forma, existindo um ponto de partida, procedeu-se à customização de cada um, em ordem a que as funcionalidades necessárias para o correcto funcionamento fosse possível.

- Página de entrada

  Na página de entrada, foi adicionada informação sobre o sistema. Permite efectuar o registo e o login, como podemos ver na figura 4.5.

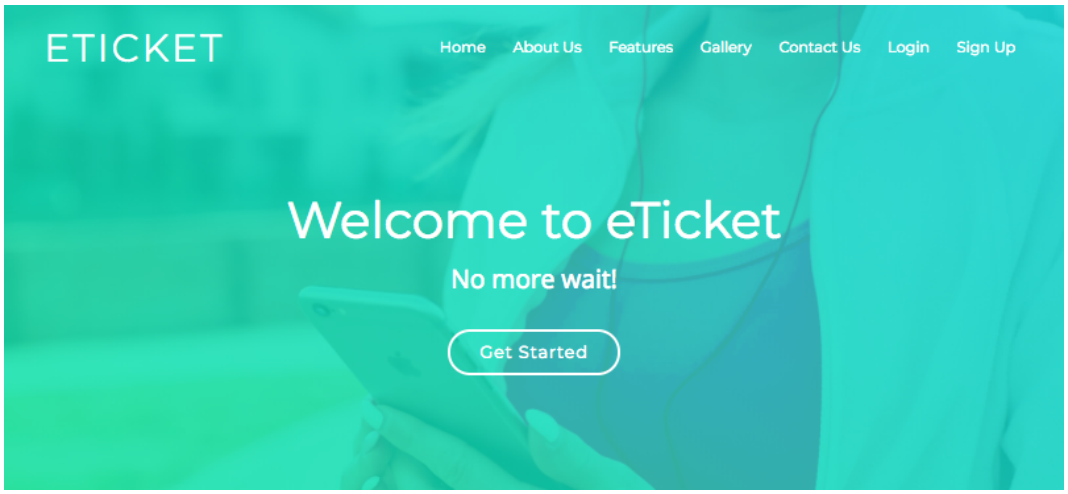

Figura 4.5: *Screenshot* da *landing page*

- Selecção de serviço de atendimento

  Através da pesquisa em mapa, é possível seleccionar o serviço de atendimento pretendido. Após a selecção de um serviço, são listados todos os balcões disponíveis tal como a informação sobre o estado do atendimento de cada um. Para cada um deles existe a opção de emissão de senha, como podemos ver na figura 4.6.

- Detalhes eTicket

  Nesta página, são listadas as informações relativas a cada um dos *eTickets* emitido para o utilizador. Podendo, em tempo real, acompanhar qual a senha em atendimento e o tempo esperado para a sua vez (ver na figura 4.7).

Após este processo de prototipagem aplicacional, foram implementadas as funcionalidades e feitas as ligações à API.

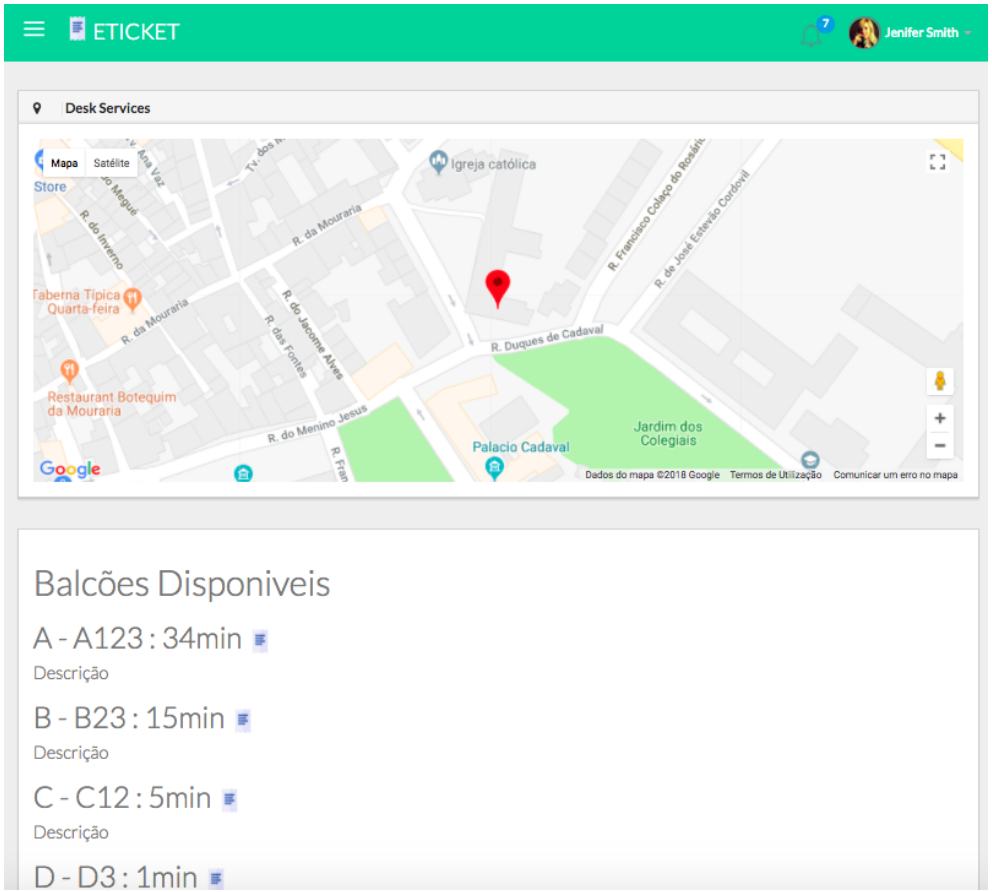

Figura 4.6: *Screenshot* da selecção de serviço de atendimento

Figura 4.7: *Screenshot* do *eTicket*

4.5 Aplicação Móvel

Após a implementação da aplicação *web*, sendo ela responsiva e de fácil utilização, o próximo passo foi convertê-la em PWA. Para tal, foram feitas as necessárias alterações. Foi adicionado o ícone da aplicação, criado e associado o Manifesto[manifest.json, 2018]

e por último adicionado o *service worker*. Para gerar o *service worker*, foi utilizado o gerador *online* PWABuilder[PWA Builder, 2018], que permite um processo assistido na sua geração. Para este projecto foi escolhida a funcionalidade "*Cache-first network*", que permite criar um sistema de versionamento da *cache* e dessa forma o seu carregamento de forma imediata, actualizando caso existam alterações no servidor em *background*. Em breve estará disponível a funcionalidade "*Advanced Pre-cache*". Esta irá permitir um ganho superior no desempenho aplicacional, pois para além do sistema *pre cache* e do funcionamento *offline*, será possível configurar a utilização da *cache* de várias formas, página a página, e ainda a disponibilização de conteúdo dinâmico mesmo que em modo *offline*.

Na figura 4.8 e 4.9, podemos ver o resultado da transformação da aplicação *web* em PWA, gerando um ícone de aplicação móvel e a sua utilização em modo *offline*.

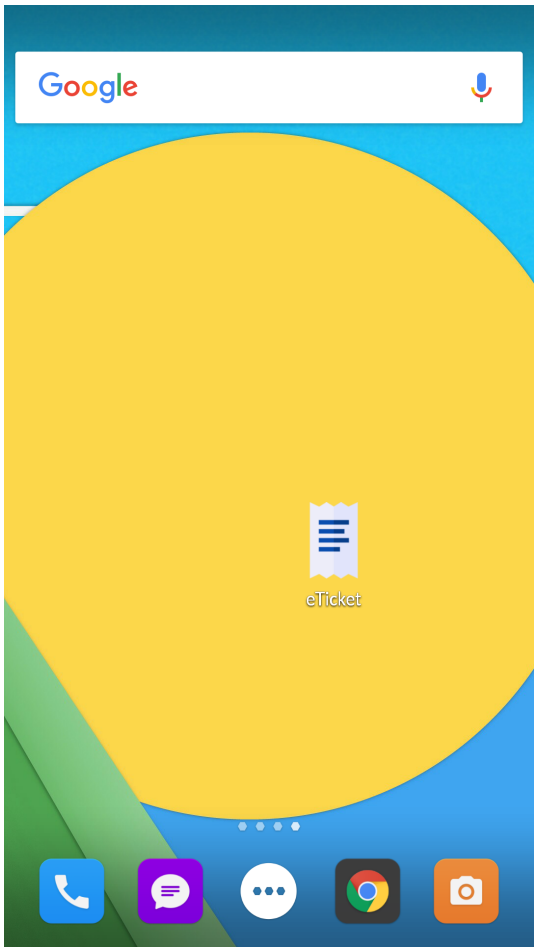

Figura 4.8: *Screenshot* do icone da aplicação

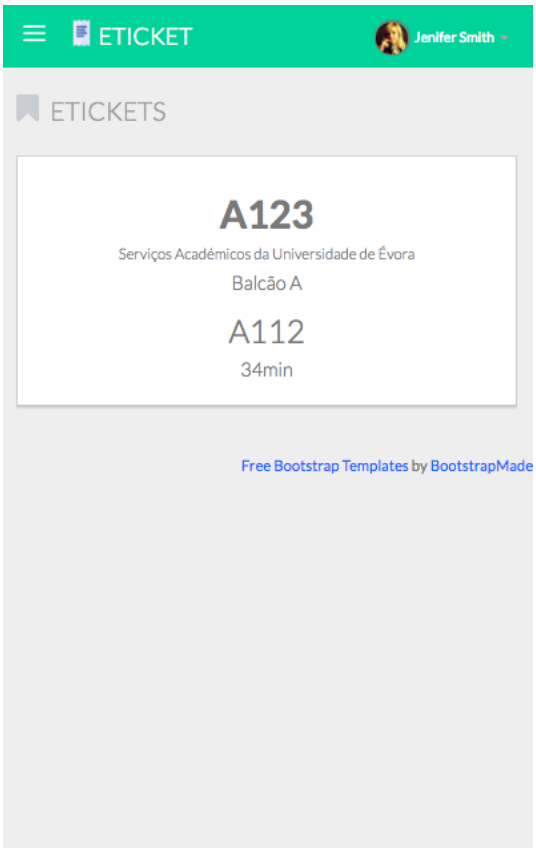

Figura 4.9: *Screenshot* do *eTicket* na versão móvel em modo *offline*

Capítulo 5

Conclusões

Neste capítulo são apresentadas as principais conclusões relativamente ao trabalho desenvolvido. São avaliados os resultados comparativamente com os objectivos inicialmente propostos. É listado o trabalho mais relevante a realizar tendo como premissa a melhoria do actual sistema.

5.1 Conclusão

De um modo geral os objectivos propostos foram atingidos. Foi possível criar um sistema de raíz, capaz de dar resposta a todos os pontos:

- Criação de sistema de *e-tickets* para serviços de atendimento - o sistema criado, permite a consulta dos vários balcões de atendimento por serviço, consulta do tempo médio de espera para cada um, a criação de senhas para um serviço de atendimento, consulta do estado actual do serviço, posição na fila de espera e tempo esperado para atendimento. Para isso, basta que o serviço esteja integrado.

- Ambiente Web[WWW, 2014] e Móvel[Computing, 2014] - Ambas as versões foram criadas. Ainda que a versão móvel não seja uma solução nativa, cumpre os requisitos.

- Complemento aos actuais sistemas de gestão de atendimento - permite a integração com os serviços existentes, bastando que o actual serviço tenha um *software* de gestão com ligação à internet e utilize os serviços *web* disponíveis em cada alteração de estado.

- Integrar sistemas idênticos, funcionando como agregador - é possível a integração

com esses sistemas, permitindo a interoperabilidade entre os vários sistemas proporcionada pela criação da API de serviços *web*.

- Dados disponíveis em tempo real[Real-time, 2014] - O sistema actual, por ser *web-based*, permite a informação dos dados em tempo real.
- Multiplataforma[Multiplataforma, 2014] - A implementação da aplicação *web* permite a utilização via *browser* em todas as plataformas.
- Apelativa e Intuitiva - A utilização de um template da *framework* Bootstrap, permitiu de forma fácil implementar um portal responsivo e visualmente evoluído. Devido também às poucas funcionalidades que possui, torna a sua experiência de utilização intuitiva.
- Tempo de espera estimado por serviço de atendimento - uma das funcionalidades já citadas, é possível através da média das diferenças dos tempos de emissão e atendimento das senhas geradas por um balcão de um determinado serviço.
- Plataforma assente num sistema altamente escalável e de grande acessibilidade - A arquitectura do sistema foi desenhada de modo a permitir a sua evolução à medida que a sua utilização for crescendo.
- Ligação à internet para actualização de dados e pedidos de *e-tickets* e posteriormente possibilidade de utilização *offline* nas aplicações móveis - A utilização dos *service workers*, através da transformação da aplicação *web* em PWA, veio tornar este cenário possível.

5.2 Trabalho Futuro

Existe muito a fazer para dar continuidade a este projecto :

- Teste piloto, implementação deste sistema, num serviço existente, por exemplo nos Serviços Académicos da Universidade de Évora. O sistema utilizado por este serviço de atendimento parece reunir as condições para ter uma fácil integração. De facto possui um *software* dedicado, com ligação à internet, e é também um serviço que tem muitas oscilações no fluxo de utentes. Seria um excelente teste.
- Implementação de aplicação móvel em React Native. Não obstante, de existir uma versão para dispositivos móveis, esta não é nativa. Todavia, não afectando muito a sua utilização, permitiria um melhor funcionamento *offline*, dos recursos do dispositivo e de notificações.
- Base de dados distribuída, seja através de um *cluster*, com a utilização de *sharding* das tabelas com mais registos, ou através da utilização de uma base de dados em NoSQL para as mesmas tabelas. Manter-se-ia o resto do sistema relacional na base de dados actual.

- Criação de posto de emissão de senhas no local do serviço de atendimento, para substituição dos sistemas analógicos, permitindo integração com o sistema actual.

5.3 Considerações Finais

Chegados ao fim, resta-nos assumir, nesta breve nota final, uma sensação ambivalente, aparentemente contraditória. Em termos imediatos percorre-me um respirar de alívio, ao fim de um aturado trabalho, de dedicação exaustiva, presença constante nos diferentes tabuleiros onde a minha vida se desenrola: família, empresa, universidade. Cedo a ideia surgiu, germinou, desenvolveu-se, foi acarinhada, estimulada pelos diferentes agentes a quem foi apresentada, e a sua proposta aí está! Já não a assumimos como nossa! Oxalá que, breve passo que seja, possa vir a ter algum acolhimento e possa cumprir os objectivos operacionais a que se propôs. Por outro lado, temos consciência que, a partir daqui, muito haveria a desenvolver na busca de alargar o conceito e a sua aplicação a um universo de serviços mais vasto. Pronto! O que está feito aqui se patenteia! Reitero e renovo os meus agradecimentos a todos os que de uma forma ou outra, directa ou indirectamente, contribuíram de forma gentil e profissional para a sua realização.

Referências bibliográficas

[Android, 2014] Android (2014). android. http://www.android.com/. [Online; accessed 11-Jan-2014].

[API, 2006] API, W. (2006). API. http://pt.wikipedia.org/wiki/API. [Online; accessed 11-Jan-2014].

[Computing, 2014] Computing, W. M. (2014). Mobile Computing. http://en.wikipedia.org/wiki/Mobile_computing. [Online; accessed 11-Jan-2014].

[Design, 2014] Design, W. R. W. (2014). Responsive Web Design. http://en.wikipedia.org/wiki/Responsive_web_design. [Online; accessed 11-Jan-2014].

[Friendly, 2014] Friendly, W. U. (2014). User Friendly. https://pt.wiktionary.org/wiki/user_friendly. [Online; accessed 11-Jan-2014].

[Interface, 2014] Interface, W. (2014). Interface. http://pt.wikipedia.org/wiki/Interface. [Online; accessed 11-Jan-2014].

[JSON, 2014] JSON (2014). JSON. http://json.org/. [Online; accessed 11-Jan-2014].

[Layout, 2014] Layout, W. (2014). Layout. http://pt.wikipedia.org/wiki/Layout_(computa%C3%A7%C3%A3o). [Online; accessed 11-Jan-2014].

[Multiplataforma, 2014] Multiplataforma, W. (2014). Multiplataforma. http://pt.wikipedia.org/wiki/Multiplataforma. [Online; accessed 11-Jan-2014].

[MySQL, 2014] MySQL (2014). MySQL. http://www.mysql.com/. [Online; accessed 11-Jan-2014].

[Oracle, 2018] Oracle (2014). Oracle. https://www.oracle.com/database/index.html. [Online; accessed 15-Mar-2018].

[Microsoft SQL Server, 2018] Microsoft (2018). SQLServer. https://www.microsoft.com/en-us/sql-server/. [Online; accessed 15-Mar-2018].

[Navegador, 2014] Navegador, W. (2014). Navegador. https://pt.wikipedia.org/wiki/Navegador. [Online; accessed 11-Jan-2014].

[NoSQL, 2014] NoSQL, W. (2014). NoSQL. http://pt.wikipedia.org/wiki/NoSQL. [Online; accessed 11-Jan-2014].

[Offline, 2006] Offline, W. (2006). Offline. http://en.wikipedia.org/wiki/Online_and_offline. [Online; accessed 11-Jan-2014].

[C Sharp, 2018] C Sharp, M. (2018). CSHARP. https://pt.wikipedia.org/wiki/C_Sharp. [Online; accessed 15-Mar-2018].

[Java, 2018] Java, O. (2018). Java. https://www.java.com/. [Online; accessed 15-Mar-2018].

[JavaScript, 2018] JavaScript, J. (2018). JavaScript. https://www.javascript.com/. [Online; accessed 15-Mar-2018].

[PHP, 2014] PHP (2014). PHP. http://www.php.net/. [Online; accessed 11-Jan-2014].

[Real-time, 2014] Real-time, W. (2014). Real-time. http://en.wikipedia.org/wiki/Real-time. [Online; accessed 11-Jan-2014].

[WBS, 2014] WBS, W. (2014). WBS. http://en.wikipedia.org/wiki/Work_breakdown_structure. [Online; accessed 11-Jan-2014].

[WWW, 2014] WWW, W. (2014). World Wide Web. http://pt.wikipedia.org/wiki/World_Wide_Web. [Online; accessed 11-Jan-2014].

[Servidor Web, 2017] WebServer, W. (2017). Web Server. https://pt.wikipedia.org/wiki/Servidor_web. [Online; accessed 15-Mar-2018].

[Hypertext Transfer Protocol, 2018] HTTP, W. (2017). HTTP. https://pt.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Online; accessed 15-Mar-2018].

[Apache, 2018] Apache, A. (2018). Apache. https://www.apache.org/. [Online; accessed 15-Mar-2018].

[NGINX, 2018] NGINX, N. (2018). NGINX. https://www.nginx.com/. [Online; accessed 15-Mar-2018].

[Microsoft IIS, 2018] IIS, M. (2018). IIS. https://www.iis.net/. [Online; accessed 15-Mar-2018].

[Node.js, 2018] Node, N. (2018). IIS. https://nodejs.org. [Online; accessed 15-Mar-2018].

[TomCat, 2018] TomCat, A. (2018). TomCat. http://tomcat.apache.org/. [Online; accessed 15-Mar-2018].

[Lighttpd, 2018] Lighttpd, N. (2018). Lighttpd. https://www.lighttpd.net/. [Online; accessed 15-Mar-2018].

[Open-source, 2018] Open-source software, W. (2018). OpenSource. https://en.wikipedia.org/wiki/Open-source_software. [Online; accessed 15-Mar-2018].

[Load Balancing, 2018] Load balancing (computing), W. (2018). LoadBalancing. https://en.wikipedia.org/wiki/Load_balancing_(computing). [Online; accessed 15-Mar-2018].

[SOAP, 2018] SOAP, W. (2018). SOAP. https://pt.wikipedia.org/wiki/SOAP. [Online; accessed 18-Mar-2018].

[REST, 2018] REST, W. (2018). REST. https://pt.wikipedia.org/wiki/REST. [Online; accessed 18-Mar-2018].

[Swagger, 2018] Swagger, S. (2018). Swagger. https://swagger.io/. [Online; accessed 18-Mar-2018].

[GraphQL, 2018] GraphQL, G. (2018). GraphQL. http://graphql.org/. [Online; accessed 18-Mar-2018].

[Responsive, 2018] Html Responsive, W. (2018). Responsive. https://www.w3schools.com/html/html_responsive.asp. [Online; accessed 18-Mar-2018].

[Bootstrap, 2018] Bootstrap, B. (2018). Bootstrap. https://getbootstrap.com/. [Online; accessed 18-Mar-2018].

[Foundation, 2018] Foundation, Z. (2018). Foundation. https://foundation.zurb.com/. [Online; accessed 18-Mar-2018].

[ZURB, 2018] ZURB, Z. (2018). ZURB. https://zurb.com/. [Online; accessed 18-Mar-2018].

[Semantic UI, 2018] Semantic, S. (2018). Semantic. https://semantic-ui.com/. [Online; accessed 18-Mar-2018].

[Pure, 2018] Pure, S. (2018). Pure. https://purecss.io/. [Online; accessed 18-Mar-2018].

[XML, 2018] XML, W. (2018). XML. https://pt.wikipedia.org/wiki/XML. [Online; accessed 18-Mar-2018].

[CSS, 2018] CSS, W. (2018). CSS. https://www.w3schools.com/css/. [Online; accessed 18-Mar-2018].

[CSS, 2018] What is a WebView?, T. (2018). WebView. https://developer.telerik.com/featured/what-is-a-webview/. [Online; accessed 18-Mar-2018].

[Apache Cordova, 2018] Apache Cordova, A. (2018). Cordova. https://cordova.apache.org/. [Online; accessed 18-Mar-2018].

[React Native, 2018] React Native, F. (2018). ReactNative. https://facebook.github.io/react-native/. [Online; accessed 18-Mar-2018].

[PWA, 2018] Progressive Web Apps, W. (2018). PWA. https://developers.google.com/web/progressive-web-apps/. [Online; accessed 18-Mar-2018].

[Microservices, 2018] Microservices, W. (2018). Microservices. https://en.wikipedia.org/wiki/Microservices. [Online; accessed 21-Mar-2018].

[Swagger Editor, 2018] Swagger, S. (2018). Swagger. http://editor2.swagger.io/. [Online; accessed 26-Mar-2018].

[PWA Builder, 2018] PWABuilder, P. (2018). PWABuilder. https://www.pwabuilder.com. [Online; accessed 27-Mar-2018].

[manifest.json, 2018] Mozilla, M. (2018). manifestjson. https://developer.mozilla.org/pt-PT/Add-ons/WebExtensions/manifest.json. [Online; accessed 27-Mar-2018].

[Bash, 2018] Bash (Unix shell), W. (2018). Bash. https://en.wikipedia.org/wiki/Bash_(Unix_shell). [Online; accessed 27-Mar-2018].

[NiceAdmin, 2018] BootstrapMade, B. (2018). niceadmin. https://bootstrapmade.com/nice-admin-bootstrap-admin-html-template/. [Online; accessed 27-Mar-2018].

[Avilon, 2018] BootstrapMade, B. (2018). Avilon. https://bootstrapmade.com/avilon-bootstrap-landing-page-template/. [Online; accessed 27-Mar-2018].

[Portugal, 2013] Portugal, S. S. (2013). Notícia. http://www4.seg-social.pt/noticias/-/asset_publisher/9N8j/content/seguranca-social-implementa-o-sistema-de-atendimento-por-marcacao-em-braganca-leiria-
redirect=http%3A%2F%2Fwww4.seg-social.pt%2Fnoticias%3Fp_p_id%3D101_INSTANCE_9N8j%26p_p_lifecycle%3D0%26p_p_state%3Dnormal%26p_p_mode%3Dview%26p_p_col_id%3Dcolumn-1%26p_p_col_count%3D1%26_101_INSTANCE_9N8j_advancedSearch%3Dfalse%26_101_INSTANCE_9N8j_keywords%3D%26_101_INSTANCE_9N8j_delta%3D10%26_101_INSTANCE_9N8j_cur%3D4%26_101_INSTANCE_9N8j_andOperator%3Dtrue. [Online; accessed 11-Jan-2014].

[de Notícias, 2006] de Notícias, J. (2006). Tempos de espera controlados por SMS. http://www.jn.pt/PaginaInicial/Interior.aspx?content_id=566274&page=-1. [Online; accessed 11-Jan-2014].

[AMA, 2015] AMA - Agência para a Modernização Administrativa, I.P. (2015). Serviços Públicos Perto de Si. http://www.mapadocidadao.pt. [Online; accessed 11-Jan-2014].

[SIGA, 2017] Instituto de Informática, I.P. (2017). SIGA - Sistema Integrado de Gestão de Atendimento. http://siga.seg-social.pt/. [Online; accessed 22-Set-2017].

[sigaApp, 2017] Siga App. https://play.google.com/store/apps/details?id=pt.segsocial.iies.sigaapp.prod. [Online; accessed 22-Set-2017].

[CML, 2017] Câmara Municipal de Lisboa. http://www.cm-lisboa.pt/. [Online; accessed 22-Set-2017].

[CML, 2017] Loja Lisboa - Senhas Virtuais. https://play.google.com/store/apps/details?id=pt.newvision.inlinecml&hl=pt_PT. [Online; accessed 22-Set-2017].

[SONAE, 2017] SONAE - Projecto tira-vez. https://www.sonae.pt/pt/inovacao/projetos/tira-vez/. [Online; accessed 22-Set-2017].

[Continente, 2017] Continente - apicação tira-vez. https://www.continente.pt/pt-pt/public/Pages/Apps/tira-vez-continente.aspx. [Online; accessed 22-Set-2017].

[SONAE, 2017] SONAE. https://www.sonae.pt/pt/. [Online; accessed 22-Set-2017].

[w3techs, 2018] w3techs. https://w3techs.com/technologies/overview/web_server/all. [Online; accessed 15-Mar-2018].

[UpGuard, 2017] upguard. https://www.upguard.com/articles/apache-vs-nginx. [Online; accessed 15-Mar-2018].

[w3techs, 2018] w3techs. https://w3techs.com/technologies/overview/programming_language/all. [Online; accessed 15-Mar-2018].

[w3techs, 2018] w3techs. https://w3techs.com/technologies/overview/client_side_language/all. [Online; accessed 15-Mar-2018].

[DB-Engines, 2018] db-engines. https://db-engines.com/en/ranking. [Online; accessed 15-Mar-2018].

[WSDL, 2018] WSDL. https://pt.wikipedia.org/wiki/Web_Services_Description_Language. [Online; accessed 18-Mar-2018].

[REST, 2018] rest. https://www.codecademy.com/articles/what-is-rest. [Online; accessed 18-Mar-2018].

[FEF, 2018] frontendframeworks. https://www.sitepoint.com/most-popular-frontend-frameworks-compared/. [Online; accessed 18-Mar-2018].

[FEF2, 2018] frontendframeworks2. https://stackshare.io/stackups/bootstrap-vs-foundation-vs-semantic-ui. [Online; accessed 18-Mar-2018].

[PWAs, 2017] Getting Started With Progressive Web Apps (PWA). https://medium.com/codingthesmartway-com-blog/getting-started-with-progressive-web-apps-pwa-ab05bcc25bfd. [Online; accessed 18-Mar-2018].

[w3techs, 2018] Usage statistics and market share of Node.js for websites. https://w3techs.com/technologies/details/ws-nodejs/all/all. [Online; accessed 14-May-2018].

Anexos

Anexo

Anexo 1 - Arquitectura do Sistema

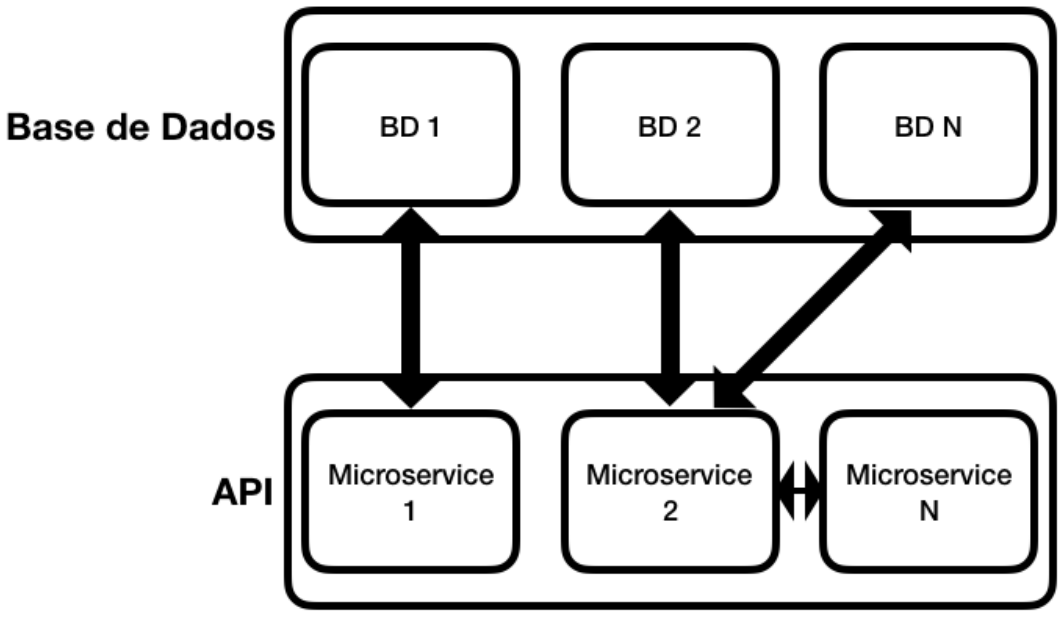

Figura 1: Exemplo de várias formas de interacção entre API constituída por microserviços e bases de dados

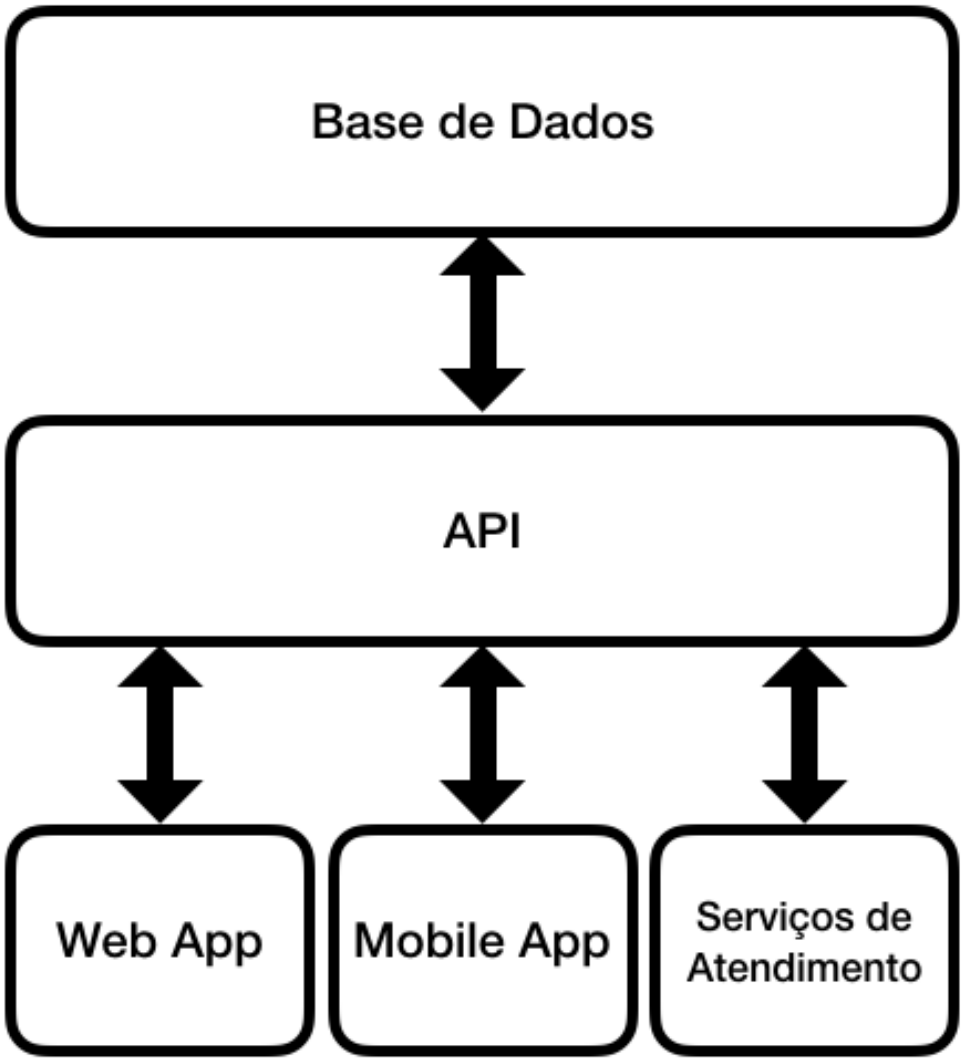

Figura 2: Arquitectura do sistema

Anexo

Anexo 2 - Base de Dados

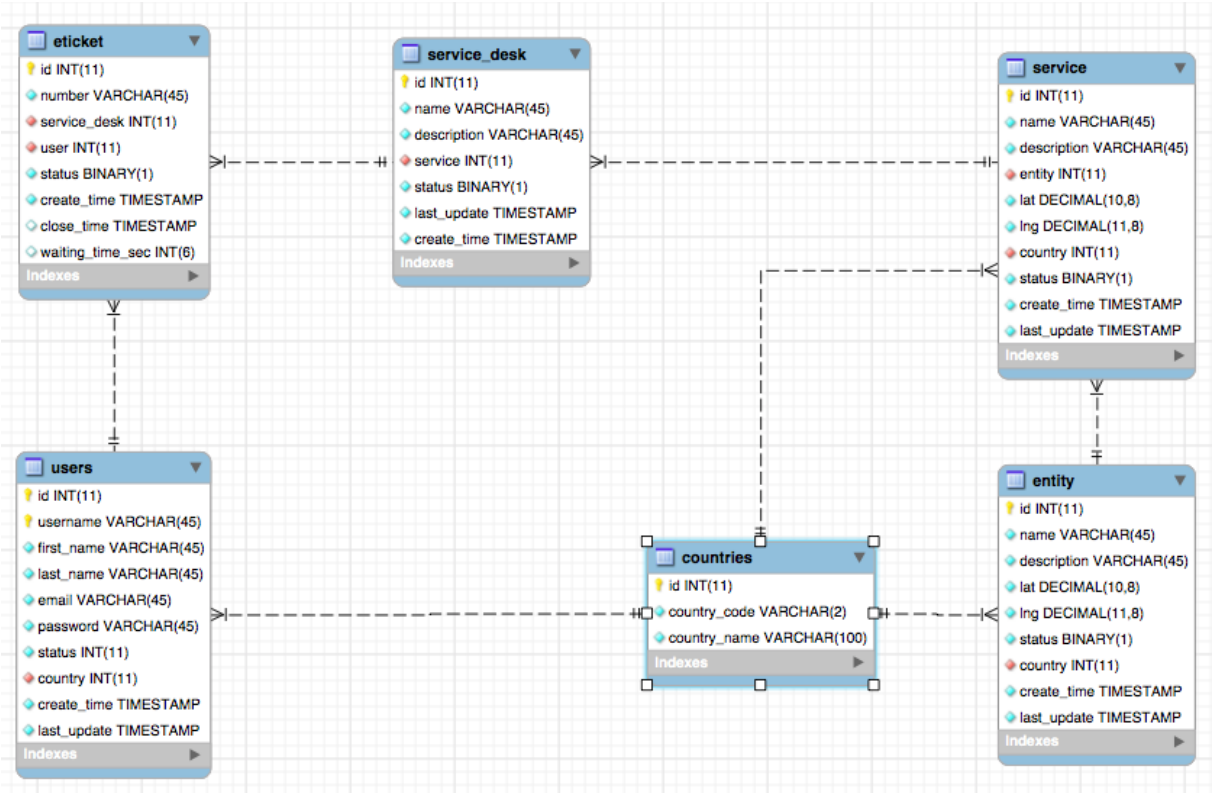

Figura 3: Representação do modelo relacional da base de dados

```
Dump da estrutura da base de dados
-- -----------------------------------------------------------------------------
-- Schema eticket
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema eticket
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `eticket` DEFAULT CHARACTER SET utf8 ;
USE `eticket` ;

-- -----------------------------------------------------
-- Table `eticket`.`countries`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `eticket`.`countries` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `country_code` VARCHAR(2) NOT NULL DEFAULT '',
  `country_name` VARCHAR(100) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `eticket`.`entity`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `eticket`.`entity` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NOT NULL,
  `entity` INT(11) NOT NULL,
  `lat` DECIMAL(10,8) NOT NULL,
  `lng` DECIMAL(11,8) NOT NULL,
  `status` BINARY(1) NOT NULL DEFAULT '0',
  `country` INT(11) NOT NULL,
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  INDEX `countries_fk3` (`country` ASC),
  CONSTRAINT `countries_fk3`
    FOREIGN KEY (`country`)
    REFERENCES `eticket`.`countries` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `eticket`.`service`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `eticket`.`service` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NOT NULL,
  `entity` INT(11) NOT NULL,
  `lat` DECIMAL(10,8) NOT NULL,
  `lng` DECIMAL(11,8) NOT NULL,
  `country` INT(11) NOT NULL,
  `status` BINARY(1) NOT NULL DEFAULT '0',
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  INDEX `entity_fk_idx` (`entity` ASC),
  INDEX `countries_fk2` (`country` ASC),
  CONSTRAINT `countries_fk2`
    FOREIGN KEY (`country`)
    REFERENCES `eticket`.`countries` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `entity_fk`
    FOREIGN KEY (`entity`)
    REFERENCES `eticket`.`entity` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `eticket`.`service_desk`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `eticket`.`service_desk` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NOT NULL,
```

```
  `service` INT(11) NOT NULL,
  `status` BINARY(1) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  INDEX `service_fk_idx` (`service` ASC),
  CONSTRAINT `service_fk`
    FOREIGN KEY (`service`)
    REFERENCES `eticket`.`service` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `eticket`.`users`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `eticket`.`users` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `status` INT(11) NOT NULL DEFAULT '0',
  `country` INT(11) NOT NULL,
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`, `username`),
  INDEX `countries_fk_idx` (`country` ASC),
  CONSTRAINT `countries_fk`
    FOREIGN KEY (`country`)
    REFERENCES `eticket`.`countries` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `eticket`.`eticket`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `eticket`.`eticket` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `number` VARCHAR(45) NOT NULL,
  `service_desk` INT(11) NOT NULL,
  `user` INT(11) NOT NULL,
  `status` BINARY(1) NOT NULL DEFAULT '0',
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `close_time` TIMESTAMP NULL DEFAULT NULL,
  `waiting_time_sec` INT(6) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `service_desk_fk` (`service_desk` ASC),
  INDEX `user_fk` (`user` ASC),
  CONSTRAINT `service_desk_fk`
    FOREIGN KEY (`service_desk`)
    REFERENCES `eticket`.`service_desk` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `user_fk`
    FOREIGN KEY (`user`)
    REFERENCES `eticket`.`users` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Anexo

Anexo 3 - API

```
--------------------------------------------------------------------------------
Excerto da estrutura para criação da documentação da API em Swagger
--------------------------------------------------------------------------------

#http://editor2.swagger.io
swagger: '2.0'
info:
  description: >-
    This is eticket API server.  You can find out more about
    eTicket at [http://eticket.com](http://eticket.com)
  version: 1.0.0
  title: eTicket API
  termsOfService: 'http://eticket.com/terms/'
  contact:
    email: info@eticket.com
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
host: api.eticket.com
basePath: /api
tags:
  - name: user
    description: Operations about user
    externalDocs:
      description: Find out more about
```

```
      url: 'http://eticket.com'
  - name: service_desk
    description: Operations about service_desk
    externalDocs:
      description: Find out more about
      url: 'http://eticket.com'
  - name: service
    description: Operations about service
    externalDocs:
      description: Find out more about
      url: 'http://eticket.com'
  - name: entity
    description: Operations about entity
    externalDocs:
      description: Find out more about
      url: 'http://eticket.com'
  - name: eticket
    description: Operations about eticket
    externalDocs:
      description: Find out more about
      url: 'http://eticket.com'
schemes:
  - http
  - https
paths:
  /user:
    post:
      tags:
        - user
      summary: Create user
      description: This can only be done by the logged in user.
      operationId: createUser
      produces:
        - application/xml
        - application/json
      parameters:
        - in: body
          name: body
          description: Created user object
          required: true
          schema:
            $ref: '#/definitions/User'
      responses:
```

```
        default:
          description: successful operation
  /user/login:
    get:
      tags:
        - user
      summary: Logs user into the system
      description: ''
      operationId: loginUser
      produces:
        - application/json
      parameters:
        - name: username
          in: query
          description: The user name for login
          required: true
          type: string
        - name: password
          in: query
          description: The password for login in clear text
          required: true
          type: string
      responses:
        '200':
          description: successful operation
          schema:
            type: string
          headers:
            X-Rate-Limit:
              type: integer
              format: int32
              description: calls per hour allowed by the user
            X-Expires-After:
              type: string
              format: date-time
              description: date in UTC when token expires
        '400':
          description: Invalid username/password supplied
...
...
...

definitions:
```

```
  User:
    type: object
    properties:
      id:
        type: integer
        format: int64
      username:
        type: string
      firstName:
        type: string
      lastName:
        type: string
      email:
        type: string
      country:
        type: integer
        format: int32
      password:
        type: string
      phone:
        type: string
      userStatus:
        type: integer
        format: int32
        description: User Status
...
...
...
  eticket:
    type: object
    properties:
      id:
        type: integer
        format: int64
      number:
        type: string
      service_desk:
        type: integer
        format: int32
      user:
        type: integer
        format: int32
      status:
```

```
        type: integer
        format: int32
        description: User Status
-------------------------------------------------------------------------------
```

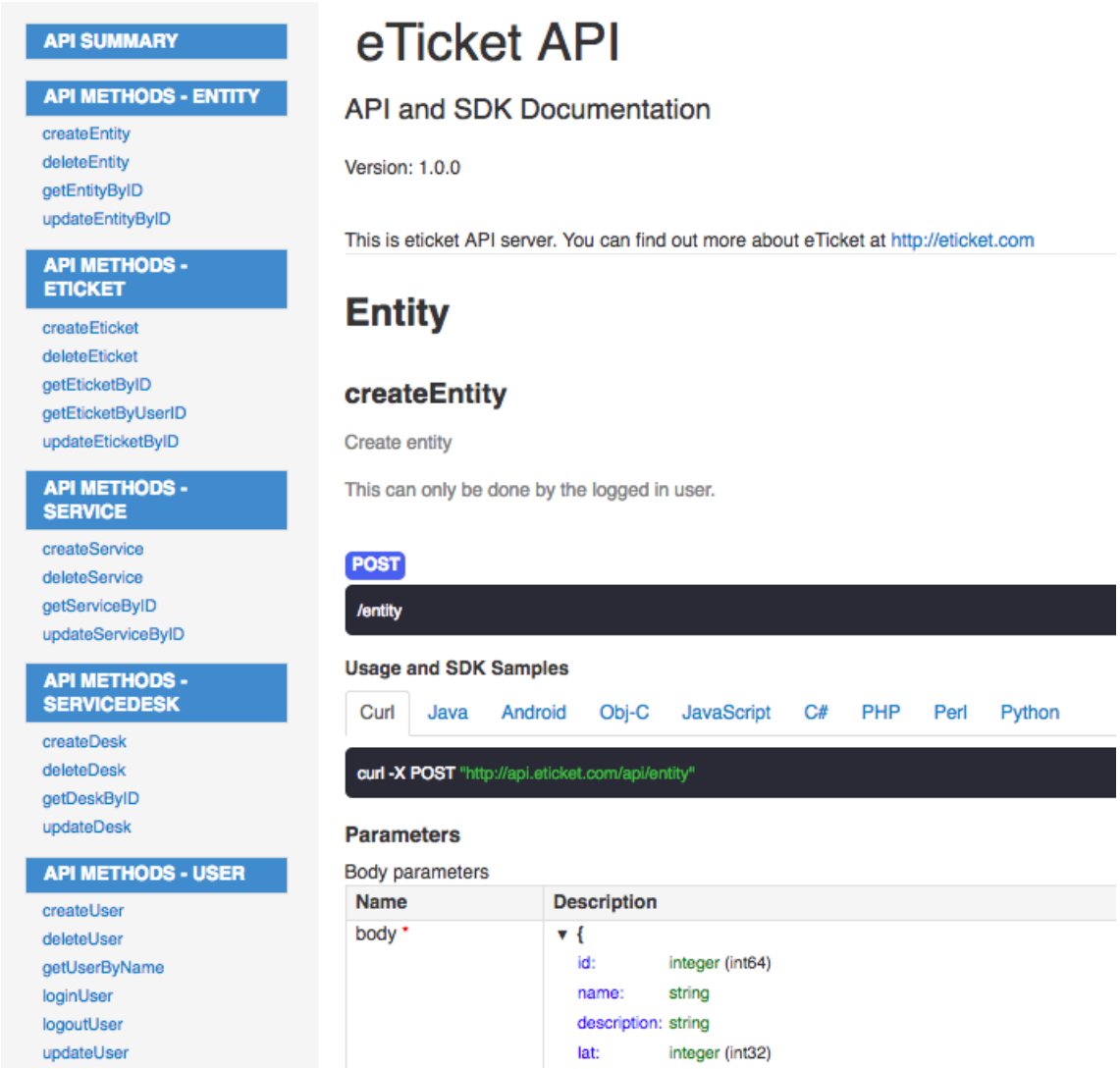

Figura 4: *Screenshot* da interface da API

Anexo

Anexo 4 - Aplicação Web

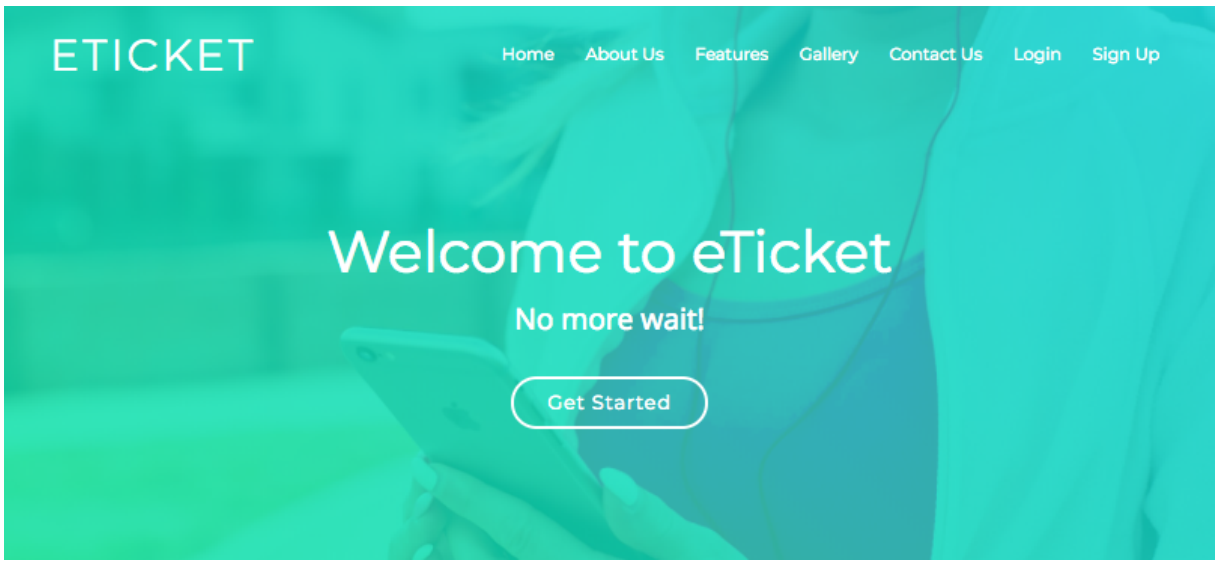

Figura 5: *Screenshot* da *landing page*

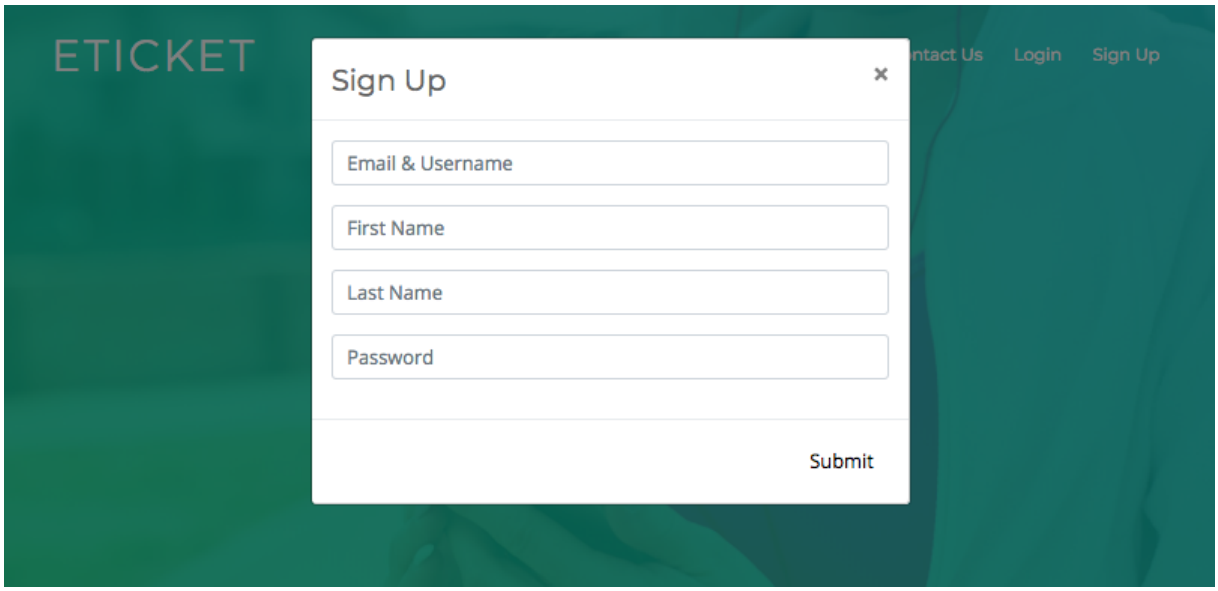

Figura 6: *Screenshot* da *Modal* de Registo

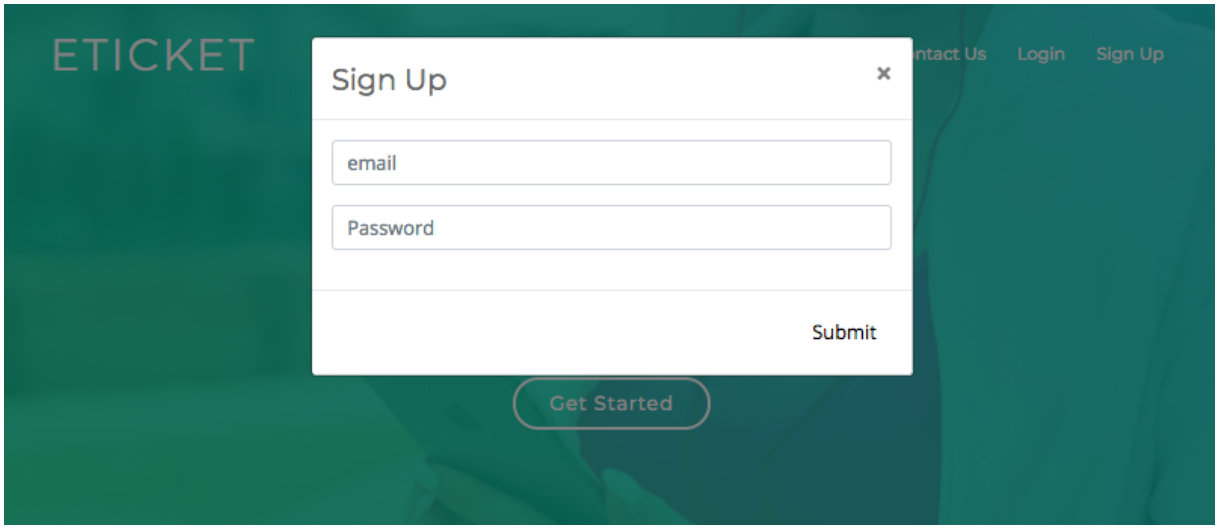

Figura 7: *Screenshot* da *Modal* de Autenticação

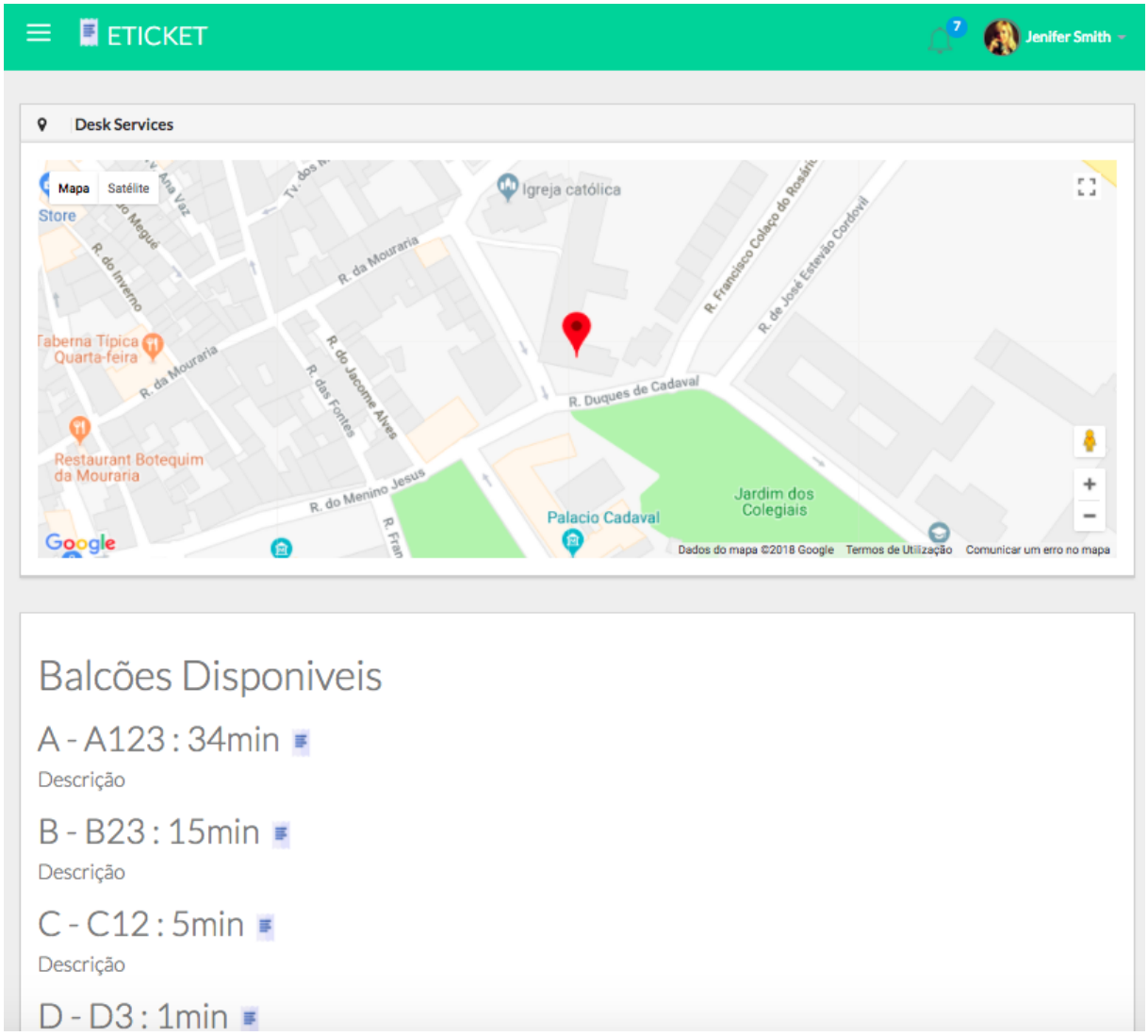

Figura 8: *Screenshot* da selecção de serviço de atendimento

Figura 9: *Screenshot* do eTicket

Anexo

Anexo 5 - Aplicação Móvel

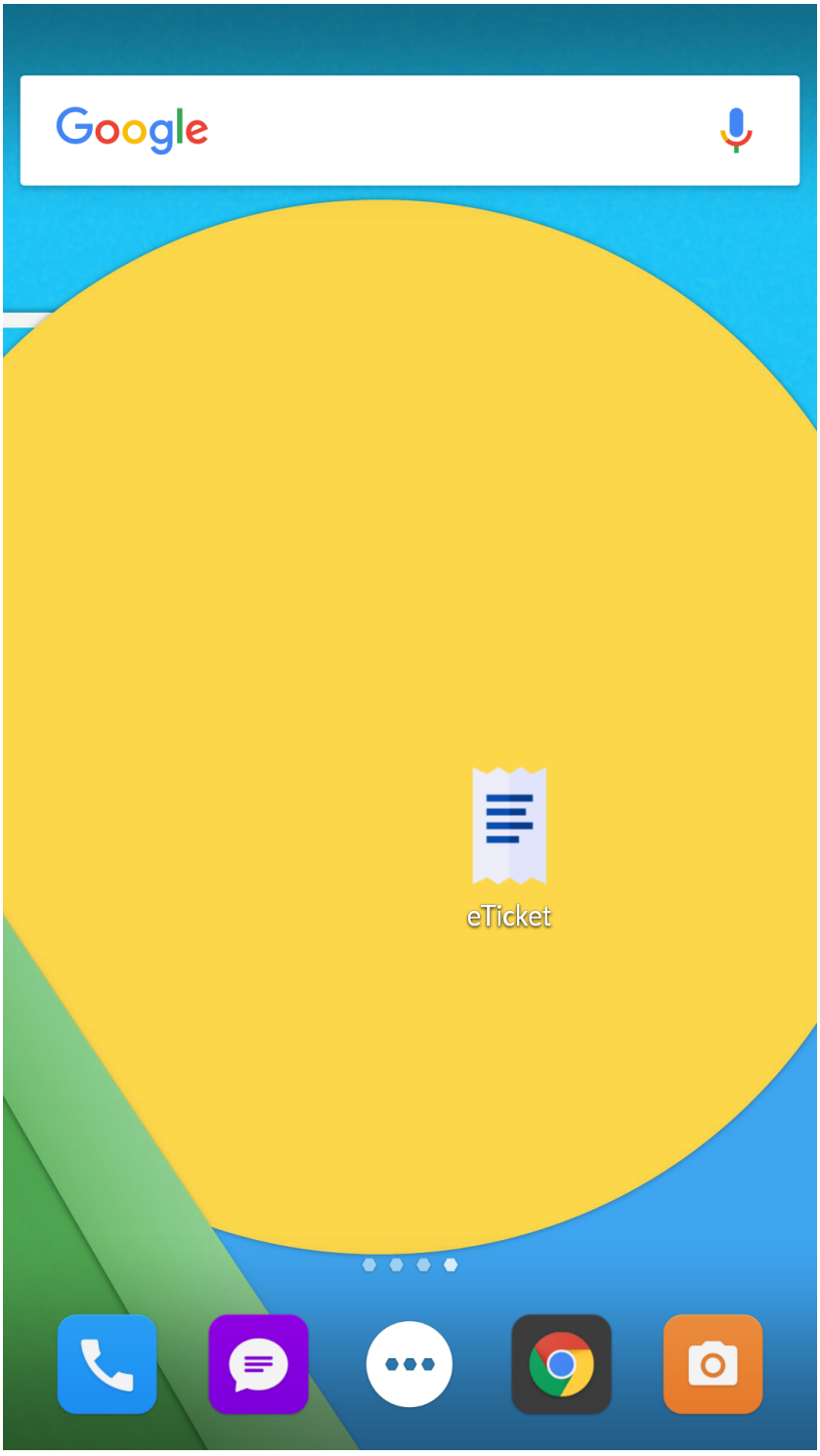

Figura 10: *Screenshot* do icon de aplicação

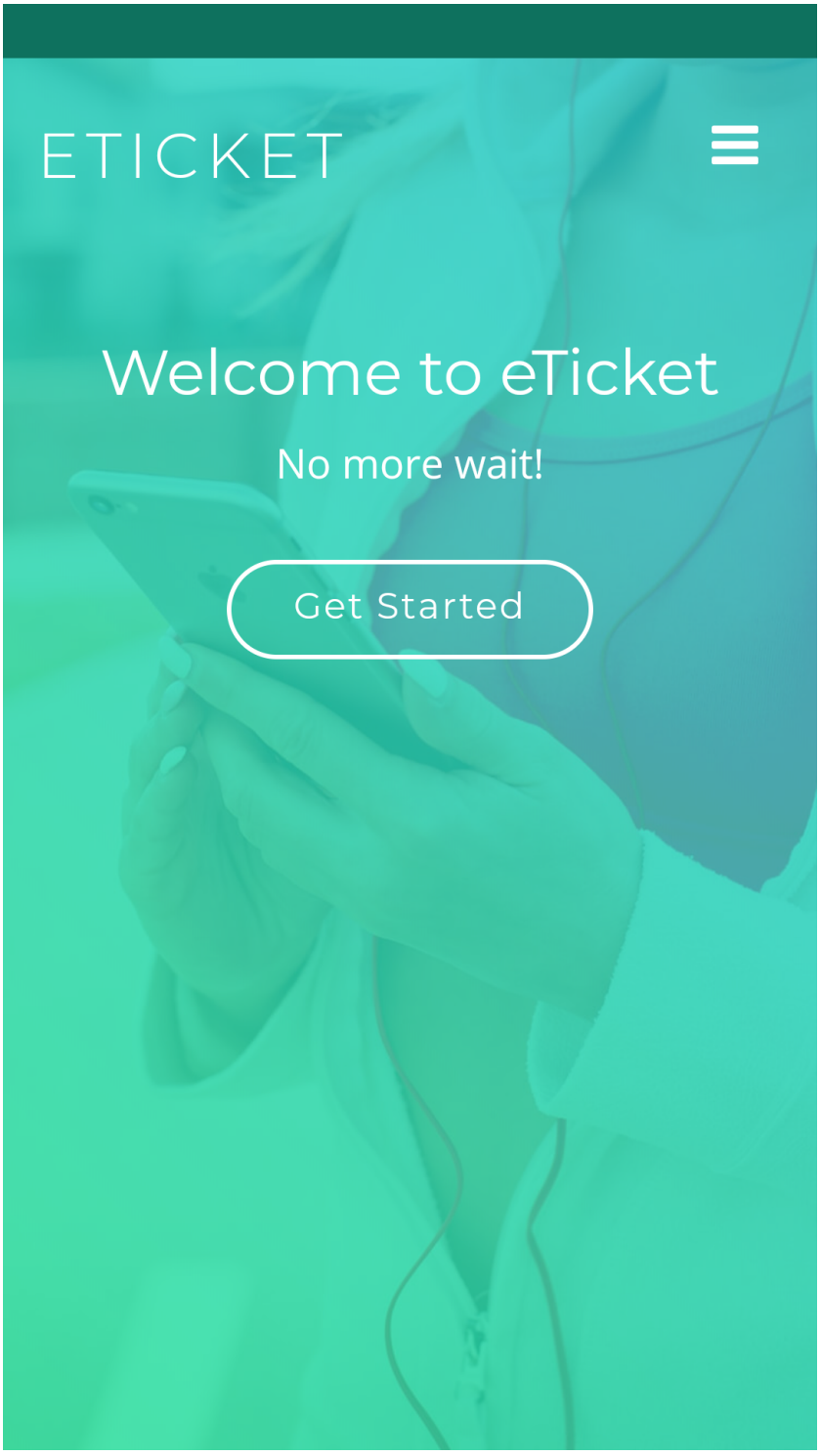

Figura 11: *Screenshot* da *landing page*

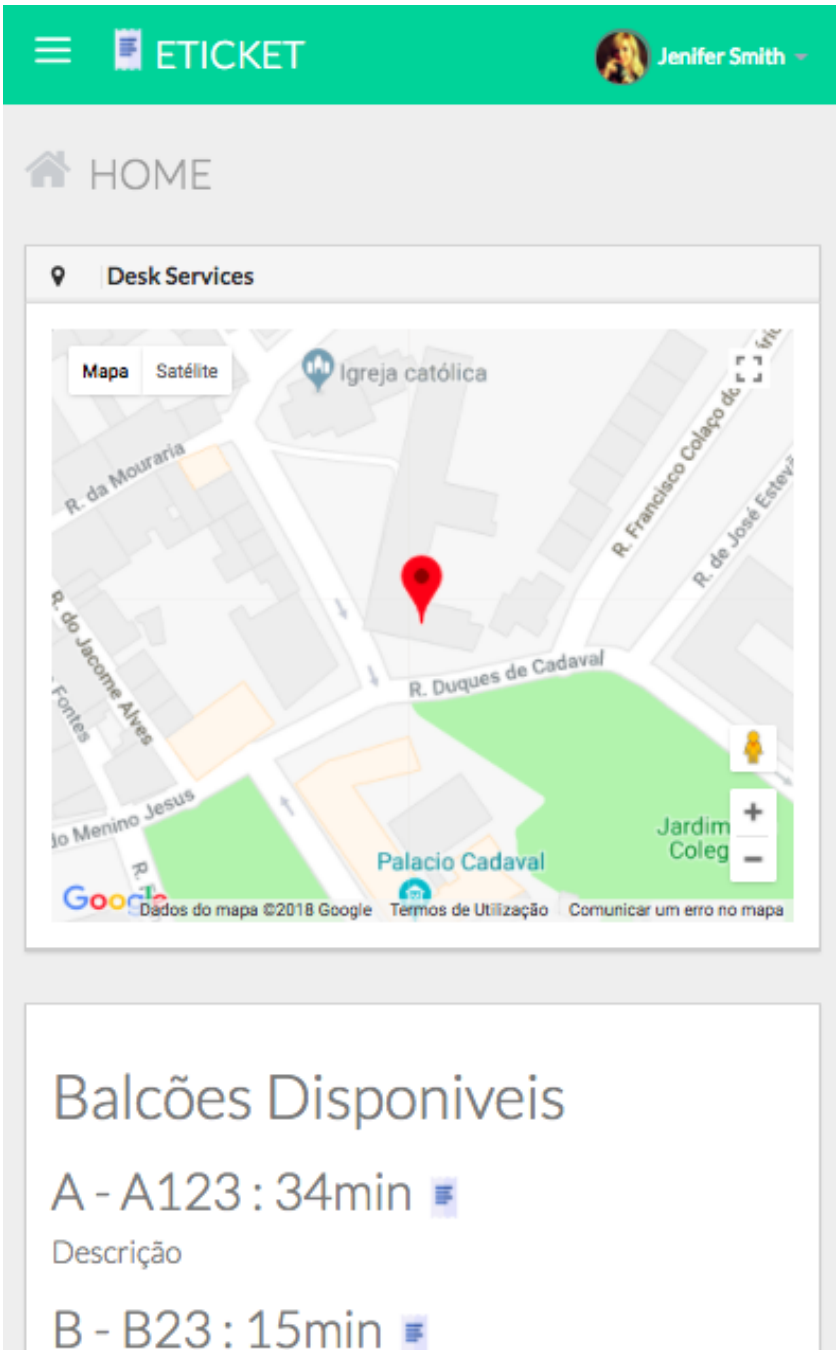

Figura 12: *Screenshot* da selecção de serviço de atendimento

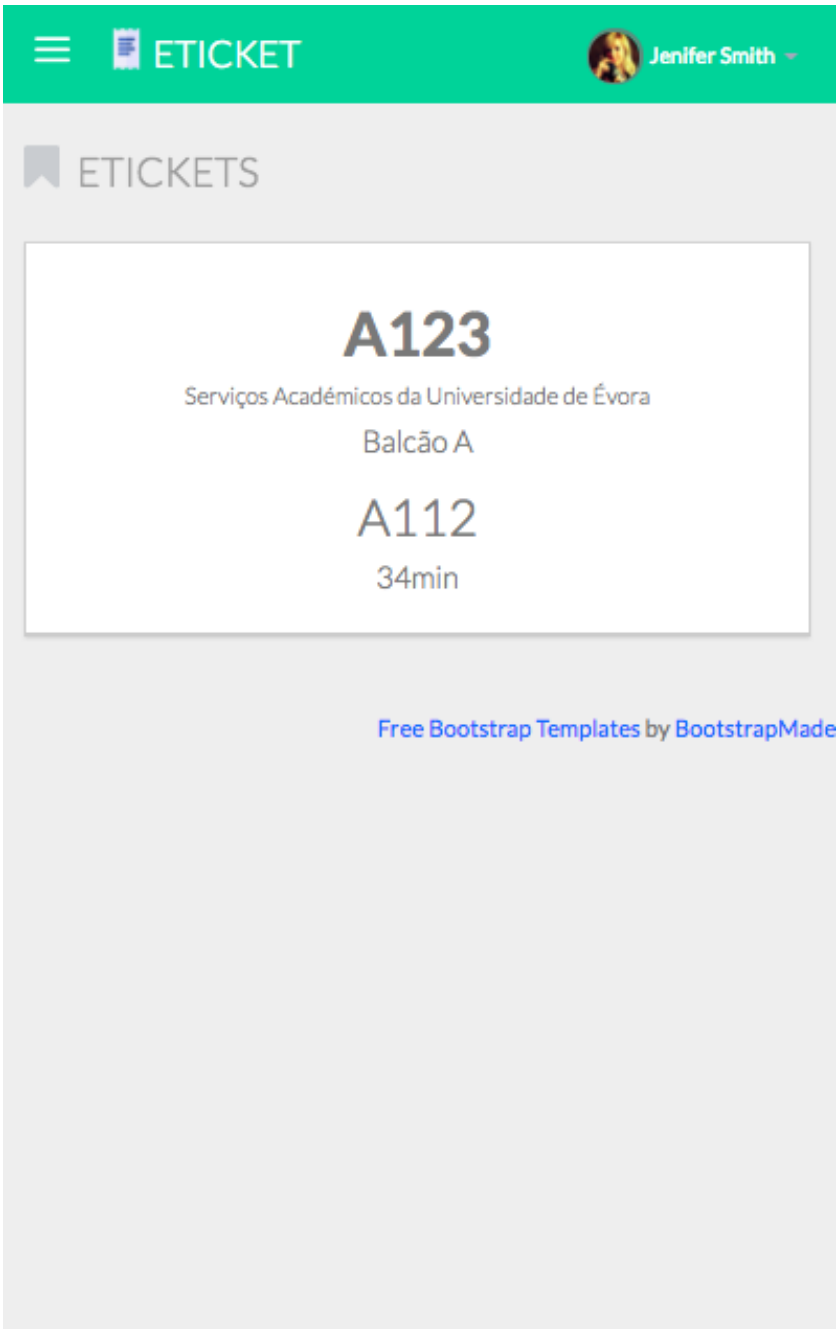

Figura 13: *Screenshot* do eTicket