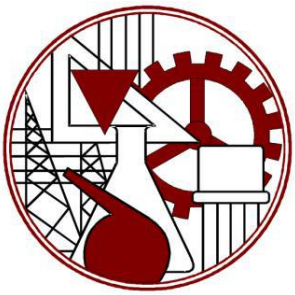
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

DEPARTAMENTO DE ENGENHARIA DE ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES



Aplicação Web de Apoio à Reciclagem de Resíduos Domésticos

André Filipe Massano Leitão

(Licenciado)

Dissertação para obtenção do grau de Mestre em Engenharia de Redes de Comunicação e Multimédia

Orientador:

Professor Doutor Paulo Manuel Trigo da Silva, ISEL, IPL

Júri:

Presidente: Professor Doutor Pedro Miguel Tores Mendes Jorge, ISEL, IPL

Vogais:

Professor Doutor Carlos Jorge de Sousa Gonçalves, ISEL, IPL

Professor Doutor Paulo Manuel Trigo da Silva, ISEL, IPL

Julho de 2021

Resumo

Vivemos uma época de grande desenvolvimento tecnológico e desgaste de recursos ambientais. Contudo, atualmente, observa-se uma crescente preocupação com o meio ambiente e respetiva adoção de comportamentos sustentáveis, em que a comunicação digital é um fator dominante. O dia-a-dia de cada consumidor é apoiado por plataformas e aplicações que promovem e auxiliam as suas decisões.

Nesse contexto, este projeto tem por objetivo desenvolver uma aplicação *web* para apoiar os consumidores nas suas ações diárias referentes à reciclagem, onde são reunidas diversas informações sobre o tema, tais como materiais recicláveis e respetiva categoria, localização dos ecopontos e o seu estado (vazio ou cheio), apoio ao reconhecimento dos vários resíduos reciclados pelo consumidor e partilha de informação relevante entre utilizadores nas redes sociais.

Desenvolveu-se uma arquitetura baseada num modelo cliente-servidor sobre o qual é possível implementar a aplicação de um modo que seja de fácil utilização e expansão futura. Nesta aplicação, houve a necessidade de integrar recursos externos permitindo enriquecê-la em temas importantes tais como, a classificação de imagens com recurso a redes profundas convolucionais, a integração com serviços de geolocalização e a interação via redes sociais.

A validação da aplicação foi feita através de testes de usabilidade onde foram questionados vários utilizadores de três faixas etárias diferentes de modo a avaliar as opções tomadas e identificar melhorias. Para garantir acesso por parte de todos os utilizadores, recorreu-se à atribuição de um domínio estando assim a aplicação permanentemente disponível para utilização.

Os testes de usabilidade tiveram excelentes resultados, classificando a aplicação como bem desenvolvida e simples de usar. Os utilizadores têm como pensamento usar frequentemente a aplicação, visto que é necessária e inovadora no contexto do tema da reciclagem.

No futuro, esta aplicação tem a possibilidade de ser aperfeiçoada e integrada com mais funcionalidades que sejam úteis aos utilizadores.

Palavras-Chave:
Aplicação *Web*, Reciclagem, Tecnologia, Consciência Ecológica, Comunicação Digital

Abstract

We live in a time of great technological development and waste of environmental resources. However, nowadays, there is a growing concern with the environment and the respective adoption of sustainable behaviours which digital communication is a dominant factor. Every consumer's daily life is supported by platforms and applications that promote and assist their decisions.

In this context, this project aims to develop a web application to support consumers in their daily actions related to recycling, where are assembled information about the topic such as recyclable materials and respective category, location of points of recycling and their status (empty or full), support of recognition of waste recycled by the consumer and share of relevant information between users that interact via social networks.

An architecture based on the client-server model was specified which it is possible to implement an application in a way to be easy to use and expand in the future. In this application, there is a need to integrate external resources allowing it to be enriched in important topics such as, image classification using deep convolution networks, integration with geolocation services and interaction with social networks.

The validation of the application was done through usability tests where several users of three different age groups were questioned to evaluate the options taken and identify improvements. In order to guarantee the access to all users, it was necessary to acquire a network domain to ensure that application is permanently available for using.

The usability tests had excellent results, classifying the application as well integrated and simple to use. Users are thinking of using the application frequently as it is necessary and innovative in the context of recycling.

In the future, this application has the possibility to be improved and integrated with more functionalities that are useful to users.

Keywords:
Web Application, Recycling, Technology, Ecological Awareness, Digital Communication

Agradecimentos

Quero agradecer ao Instituto Superior de Engenharia de Lisboa (ISEL), especialmente ao Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores (DEETC), por tudo o que me deu para obter a minha formação académica.

Ao professor Paulo Trigo, orientador deste projeto, que desde início manifestou uma grande disponibilidade e muita paciência para me motivar a nunca desistir e pelo apoio sempre prestado.

Gostaria de agradecer a todos os meus professores, tanto na Licenciatura como no Mestrado, que ao longo destes anos contribuíram para a minha valorização pessoal e profissional.

A todos os meus colegas e amigos, pelo incentivo, apoio e pelas inúmeras horas despendidas para cumprir a nossa formação académica.

Um agradecimento especial, ao meu amigo Nuno, que me apoiou sempre que estava em dificuldades e sempre se prestou a ajudar-me e a partilhar todo o seu conhecimento para atingir e enriquecer o meu projeto.

Para finalizar, um agradecimento aos meus pais e à minha família pela paciência, incentivo e apoio dado ao longo desta etapa académica, nunca permitindo que desistisse de enfrentar todas as adversidades.

Um agradecimento muito especial à minha namorada Ana, por toda a paciência demostrada, pelo apoio incondicional, pelos incentivos e por estar sempre presente com uma palavra amiga. Ela foi fundamental para ultrapassar muitas das dificuldades que surgiram ao longo da realização deste projeto.

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

Índice de Códigos

Lista de Acrónimos

AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming Interface*
CNN	*Convolutional Neural Network*
CSS	*Cascading Style Sheets*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JDBC	*Java DataBase Connectivity*
JS	*JavaScript*
JSON	*JavaScript Object Notation*
JSP	*JavaServer Pages*
MVC	*Model-View-Controller*
POM	*Project Object Model*
REST	*REpresentational State Transfer*
RSS	*Really Simple Syndication*
SDK	*Software Development Kit*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SUS	*System Usability Scale*
TLS	*Transport Layer Security*
WSDL	*Web Services Description Language*
WWW	*World Wibe Web*
XML	*Extensible Markup Language*

Capítulo 1

Introdução

Ao longo do tempo a sociedade tem vindo a aumentar o seu consumo. A realidade em que vivemos é a de um consumismo excessivo a todos os níveis. Todos os dias cada pessoa produz cerca de 1,32 kg de lixo, o que contribui para um total de 4,75 milhões de toneladas de resíduos urbanos, e apenas 16,5% é separado para reciclagem [1].

A publicidade e o marketing têm, entre outros, o objetivo de "gerar necessidade". Isto consiste em incentivar o cidadão a procurar e adquirir cada vez mais produtos sem que a maior parte das vezes exista essa necessidade. Consequentemente, maior consumo gera mais lixo (resíduos em excesso), afetando o meio ambiente, pois escasseia o espaço para armazenar a imensa e sempre crescente quantidade de lixo.

Neste cenário, surgiu a preocupação sobre o lixo excessivo produzido e o entendimento de que este poderia ter uma nova vida através de processos de reciclagem. O passo seguinte foi explicar, detalhadamente às pessoas, o que acontece ao lixo que produzimos e quais os resíduos recicláveis ou não. Ao tornar evidente a produtores e consumidores que, por exemplo, uma embalagem pode ser reciclada ao invés de se produzir uma nova utilizando novos recursos, todos saímos a ganhar. Foram assim criadas infraestruturas, denominadas por ecopontos (pontos de recolha ecológica de lixo), que permitem a separação e o depósito do lixo visando a sua reciclagem.

Ao longo do tempo tem aumentado a consciência sobre a importância dos ecopontos e com o passar dos anos verifica-se um aumento dos volumes da reciclagem. Existem, hoje, os ecopontos mais comuns (papel/cartão, plástico/metal, vidro) mas foram, também, surgindo ecopontos para outros tipos de lixo tais como, baterias, óleos, tinteiros ou medicamentos, os quais têm ainda pouca divulgação e a sua localização é reduzida face aos três ecopontos mais usuais.

O processo de reciclagem só faz sentido se todos nós (comunidade) aderirmos em massa e com a plena consciência de que é um trabalho contínuo e que desde o momento em que a introduzimos na nossa vida é para seguir como mais uma rotina diária. Criar esta consciência em cada cidadão para fazer uma separação correta dos resíduos em vez de os colocar todos no lixo indiferenciado é um trabalho difícil e continuado sendo necessário ir adaptando aos meios de comunicação que existem. Se anteriormente o que faria mais sentido era a promoção da reciclagem através de panfletos e anúncios televisivos, atualmente com o uso diário da Internet

faz mais sentido utilizar as aplicações e *sites* para informar e incentivar a população para fazer uma separação por categorias de forma mais adequada.

O desenvolvimento de aplicações para a *World Wide Web* (também conhecida apenas por *Web)* é realizado de diferentes formas, umas mais complexas que outras, sendo um dos principais objetivos garantir que esse desenvolvimento seja realizado de forma eficiente, de modo a dar origem a aplicações disponíveis no espaço do "mundo global".

O presente trabalho consistiu na criação de uma aplicação *web* para promover e facilitar a reciclagem ao cidadão. A aplicação denomina-se por *EcoApp*. Esta escolha teve por base o tema do projeto referente à ecologia e à forma como é desenvolvido, sendo este nome o que se considerou como mais apelativo e que melhor caracteriza a aplicação.

Este trabalho foi realizado em torno dos seguintes três objetivos gerais:

- Apoio ao utilizador que pretende reciclar;

Este objetivo é o de apoiar o utilizador na reciclagem de material. Um exemplo de apoio ao utilizador consiste em saber a distância a que está de um ponto de reciclagem mais próximo onde pretende reciclar determinado resíduo e o estado do ecoponto em tempo real através da interação de utilizadores. Esta interação é feita através de um mapa que localiza os ecopontos da cidade de Lisboa. O fato de existir um incentivo regulado através de um sistema de pontuação, que resulta em descontos em diversas lojas, é uma abordagem de marketing para que o utilizador se sinta mais motivado e apoiado no uso da aplicação.

- Literacia no tema da reciclagem;

A aplicação serve de instrumento à aprendizagem do tema da reciclagem, evidenciando as regras e condições que devem existir. Fornecer uma literacia sobre o tema inclui falar sobre o que é a reciclagem, a forma como deve ser feita, o que é ou não reciclável e onde devem ser depositados os resíduos. É importante também a interpretação dos símbolos presentes nas embalagens para o consumidor fazer uma compra mais consciente. O objetivo é o de sensibilizar os utilizadores a reciclar e saber como reciclar os seus resíduos. A informação sobre este tema é importante de forma a identificar os "comportamentos ecológicos" que os cidadãos podem optar.

- Comunicação e disseminação da consciência ecológica;

Esta aplicação serve como ferramenta de comunicação que agrega as informações necessárias para promover a consciência ecológica de cada utilizador. Essa agregação é efetuada interagindo com as redes sociais existentes atualmente, tais como o *Facebook* e o *Twitter*. Essa forma de comunicação ajuda à promoção da aplicação desenvolvida e informa sobre eventos e partilha de artigos sobre o meio ambiente e sua consciencialização. A informação sobre os conselhos para viver de uma forma ecológica e a aprendizagem sobre a reciclagem no dia-a-dia são um dos meios que fazem com que o utilizador seja incentivado ao "comportamento ecológico".

Os objetivos específicos da aplicação desenvolvida são os seguintes:

- incentivar uma reciclagem realizada de forma correta;
- promover a integração numa "comunidade ecológica" através dos meios sociais, para partilha de informação;
- oferecer suporte à formulação de incentivos para que a separação ecológica cresça ainda mais;
- informar sobre as regras de separação corretas dos resíduos comuns e sua localização;
- informar, em tempo real, sobre o estado dos ecopontos aos utilizadores.

Este documento tem como estrutura um primeiro capítulo, onde são expostas as motivações para a realização do projeto, os objetivos a alcançar com a realização da aplicação *web* e em que consiste a sua realização. No segundo capítulo, é apresentado o contexto em que este projeto se insere, onde são definidos claramente os aspetos diferenciadores do projeto para com outras aplicações *web* já existentes na Internet e, em que são comparadas as funcionalidades entre elas. No terceiro capítulo, inicia-se a apresentação do projeto de uma forma mais detalhada, onde são apresentados os fundamentos principais para a realização do modelo proposto e seu enquadramento tecnológico. Em seguida, são definidos os requisitos funcionais e não funcionais do modelo, e por fim, os métodos, os modelos e as formulações que foram abordadas para o desenvolvimento do projeto. O quarto capítulo consiste na exposição dos aspetos de implementação do modelo que foi proposto nos capítulos anteriores que tem como objetivo identificar as opções teóricas e justificar todas a dependências tecnológicas assumidas no projeto. O detalhe do desenvolvimento da *EcoApp* é apresentado neste capítulo baseando-se nos dois capítulos anteriores. O quinto capítulo apresenta todas as validações e testes da aplicação feitos por pessoas para avaliar o projeto desenvolvido. Todas as avaliações vão ser mostradas e serão retiradas as devidas conclusões a todos os testes realizados. Por fim, o sexto capítulo apresenta as conclusões que se obtiveram durante a realização do projeto e descreve o trabalho futuro que pode ser realizado de forma a tornar a aplicação *web* desenvolvida ainda mais rica.

Capítulo 2

Trabalho Relacionado

Neste capítulo são analisadas abordagens ao desenvolvimento de aplicações *web*, tendo sido selecionadas as que refletem as principais tendências atuais. O objetivo é o de enquadrar este projeto no contexto de outras abordagens e aplicações focadas no tema da reciclagem.

O presente projeto foi desenvolvido na sequência da dissertação de Mestrado com o nome de "Conceção de uma aplicação móvel no âmbito da ecologia e sustentabilidade ambiental" do curso Audiovisual e Multimédia [2]. Esse trabalho teve como objetivo o desenho de uma interface gráfica suportada nos fundamentos teóricos sobre a reciclagem e de desenho de interfaces gráficas, mas sem qualquer tipo de desenvolvimento tecnológico, apenas usando uma ferramenta para desenho gráfico *Invision* [3]. A esquematização da interface gráfica permitiu perceber e inspirar o desenvolvimento da *EcoApp* aproveitando todas as páginas desenhadas no projeto e sobre elas desenvolvidas as funcionalidades. Em síntese, a base da interface gráfica e a arquitetura do desenho gráfico em termos de páginas disponíveis e sua ligação segue de perto a especificação inicialmente desenvolvida no projeto.

Alguns *layouts* e formatos de figuras não foram implementados devido ao fato de terem sido projetados pela ferramenta de desenho gráfico, que a nível tecnológico não é possível desenvolver da mesma forma, sendo que nesses casos, foi usado outro tipo de abordagem gráfica. No trabalho inicial, os textos do desenho gráfico figuravam como meramente ilustrativos tendo, neste trabalho, sido desenvolvidos conteúdos concretos e adequados aos temas do projeto.

Atualmente, existem plataformas que oferecem uma abordagem consolidada para construir aplicações *web* dinâmicas. Adicionalmente, existem plataformas desenvolvidas (no passado recente) sobre o tema da reciclagem que se podem comparar à *EcoApp*.

Foram analisadas algumas aplicações *web* com destaque para as seguintes:

- ***WASTEapp*** [4]: aplicação criada pela Quercus numa parceria com a Fundação Vodafone Portugal [5];
- ***RecycleNation*** [6]: aplicação criada pela empresa ERI [7], em utilização nos Estados Unidos da América.

Considerando estas duas aplicações, é possível enquadrar as funcionalidades inovadoras e diferenciadores do projeto desenvolvido. A Tabela 1 apresenta as funcionalidades disponíveis em cada aplicação.

Tabela 1 - Funcionalidades aplicações Web

Funcionalidades\Aplicações	*EcoApp*	*WASTEapp*	*RecycleNation*
Informar sobre as regras de separação	**Sim**	Sim	Não
Localização dos Ecopontos	**Sim**	Não	Sim
Distância dos Ecopontos e seu estado em tempo real	**Sim**	Não	Não
Soluções que promovam a reciclagem	**Sim**	Não	Sim
Incentivos ao cidadão pela reciclagem feita	**Sim**	Não	Não
Partilha de informação através das redes sociais	**Sim**	Sim	Sim
Notícias atualizadas sobre o tema reciclagem	**Não**	Não	Sim
Localização dos espaços de execução do processo de reciclagem	**Não**	Sim	Não
Calendário interativo dos eventos sobre o meio ambiente	**Sim**	Não	Não

As aplicações desenvolvidas têm diferenças importantes em termos do número de funcionalidades, sendo que a *WASTEapp* é a que contém menos funcionalidades.

A aplicação *WASTEapp* foi criada com o objetivo de ajudar a separar melhor o lixo. Para além do lixo doméstico comum, como as embalagens e garrafas, a aplicação *WASTEapp* responde ainda a dúvidas sobre a forma de reciclar outros objetos incluindo 50 tipos de resíduos diferentes, entre os quais as cápsulas de café, baterias de automóveis ou termómetros e outros objetos com mercúrio. A *WASTEapp* indica a localização dos Ecocentros[1] mais próximos em vez dos ecopontos como nas outras aplicações. Nesta aplicação, é permitido ao utilizador ajudar a melhorar os seus dados contribuindo com informação adequada [8].

[1]**Ecocentro,** é uma instalação onde o cidadão pode colocar resíduos recicláveis que não devem ser colocados nos caixotes de lixo comuns nem nos ecopontos. Trata-se de um espaço aberto ao público, onde é possível colocar, sem custos, diferentes tipos de resíduos em contentores próprios [9].

Figura 1 - Ambiente gráfico da WASTEapp

Na Figura 1 verifica-se que a aplicação permite ao utilizador a escolha do resíduo que quer deitar fora localizando os Ecocentros em que este resíduo se pode reciclar. A Figura 2 mostra a partilha de informação com o cidadão para sensibilizá-lo para as regras de separação.

Figura 2 - Ambiente gráfico de separação entre ecopontos

A aplicação *RecycleNation* tem como objetivo, à semelhança da *WASTEapp*, informar os cidadãos sobre o tema da reciclagem e promovê-lo para termos um meio ambiente mais saudável. Esta aplicação tem uma base de dados de reciclagem muito abrangente que ajuda a determinar o que pode ou não ser reciclado. Ela fornece aos seus utilizadores um mapa interativo com os locais de reciclagem com horários de funcionamento e informações de contato [10].

A *RecycleNation* tem a capacidade de informar os utilizadores de tudo o que se passa no mundo ambiental atualizando-os com as notícias mais recentes. A Figura 3 mostra a página principal da *RecycleNation* onde é possível pesquisar determinado resíduo num código postal específico retornando assim os pontos de recolha disponíveis para aquele resíduo na localização inserida. Na Figura 4 são mostradas as notícias atualizadas sobre o tema da reciclagem numa das páginas da aplicação *web RecycleNation.*

O sistema de localização em todas as aplicações é um fator comum, mas, diferem nos pontos de recolha dos resíduos. A *EcoApp* apresenta os ecopontos em que se pode colocar os resíduos, a *WASTEapp* mostra os Ecocentros onde pode ser recolhido e reciclado o respetivo resíduo e a *RecycleNation* apresenta pontos de reciclagem semelhantes com os ecopontos em Portugal.

Atualmente, a partilha de informação através das redes sociais é fundamental para todos os temas da nossa sociedade, por isso, estas três aplicações *web* fazem o desenvolvimento disso.

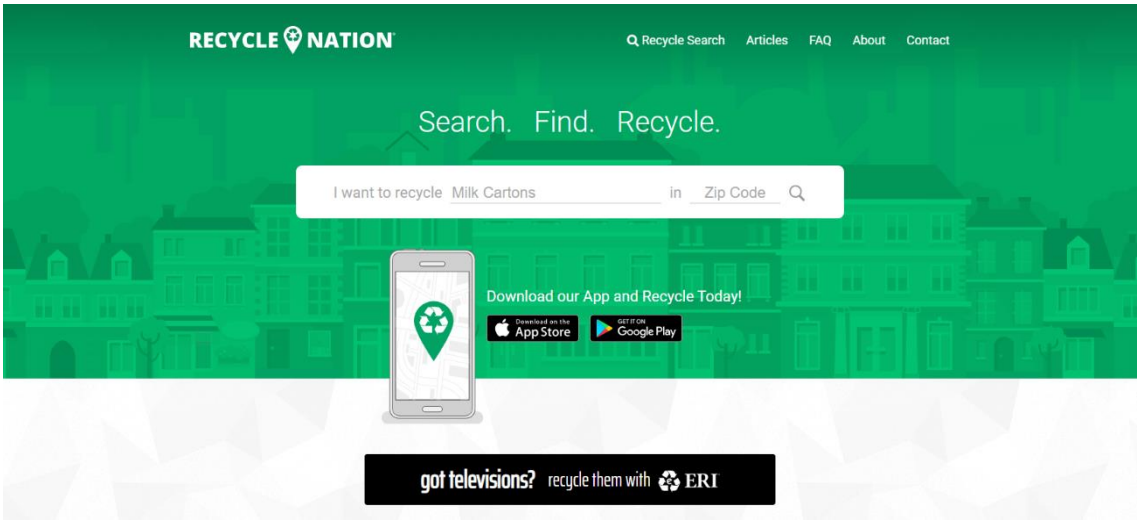


Figura 3 - Ambiente gráfico da RecycleNation

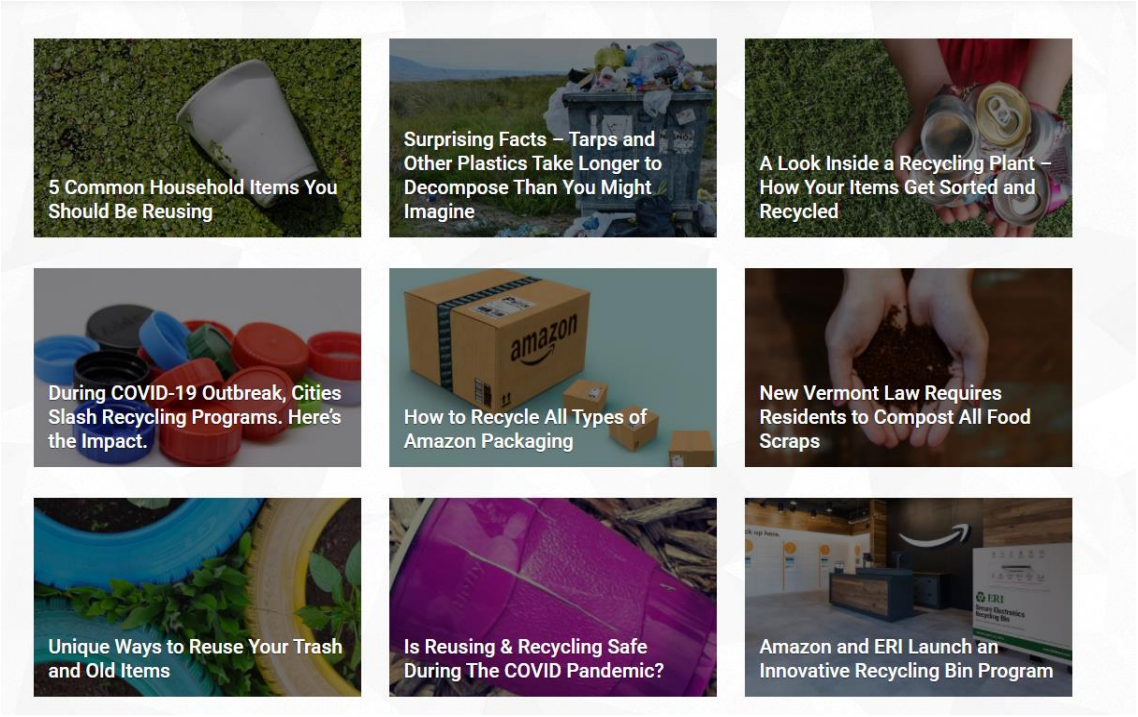


Figura 4 - Ambiente gráfico da RecycleNation (notícias)

A diferença mais relevante entre as aplicações analisadas e o projeto desenvolvido consiste na arquitetura, visto que devido a um maior número de funcionalidades implementadas na aplicação do projeto, que abrangem vários conceitos informáticos, exige-se o uso de diferentes tecnologias que aumentam a complexidade da arquitetura implementada.

Noutro ponto tecnológico, à exceção da *EcoApp*, as outras aplicações foram, também, desenvolvidas para *Android*.

Capítulo 3

Modelo Proposto

Este capítulo descreve o modelo proposto para a aplicação deste projeto, detalhadamente, os fundamentos na seção 3.1, os requisitos funcionais e não funcionais na seção 3.2, os casos de utilização na seção 3.3 e a abordagem adotada para o desenvolvimento do sistema na seção 3.4.

3.1 Fundamentos

Neste capítulo, são abordados os fundamentos teóricos aplicados no desenvolvimento da aplicação, nomeadamente, a arquitetura *web* na seção 3.1.1, o enquadramento tecnológico na seção 3.1.2, os *Web Services* na seção 3.1.3, o sistema de geolocalização na seção 3.1.4, o conceito do servidor de email na seção 3.1.5, a criptografia na seção 3.1.6 e as redes neuronais na seção 3.1.7.

3.1.1 Arquitetura *Web*

A arquitetura de uma aplicação identifica os componentes e descreve como eles se integram e estão organizados.

Um dos fatores mais importantes no desenvolvimento do projeto consiste na elaboração de uma arquitetura capaz de contemplar os seguintes critérios:

- **Simples**: Tendo em conta o tipo de utilizadores a aceder à aplicação é importante desenvolver uma aplicação e as suas interfaces o mais intuitivas possível;
- **Flexível**: Ser possível ajustar facilmente ou alterar a configuração da aplicação e ao mesmo tempo responder com rapidez e qualidade;
- **Eficaz**: A aplicação deverá ter uma grande eficiência operacional mantendo o nível de confiança dos utilizadores;
- **Escalável**: Em termos de desenvolvimento da aplicação deverá ser capaz de evoluir de uma forma fácil e estruturada.

A arquitetura lógica da aplicação *EcoApp* está dividida em camadas usando o padrão *MVC* (cf., Anexo A). A separação em camadas funcionais torna mais fácil a compreensão e modificação de cada camada A arquitetura física da aplicação baseia-se num modelo genérico cliente-servidor.

Na Figura 5 é apresentada a arquitetura geral do sistema.

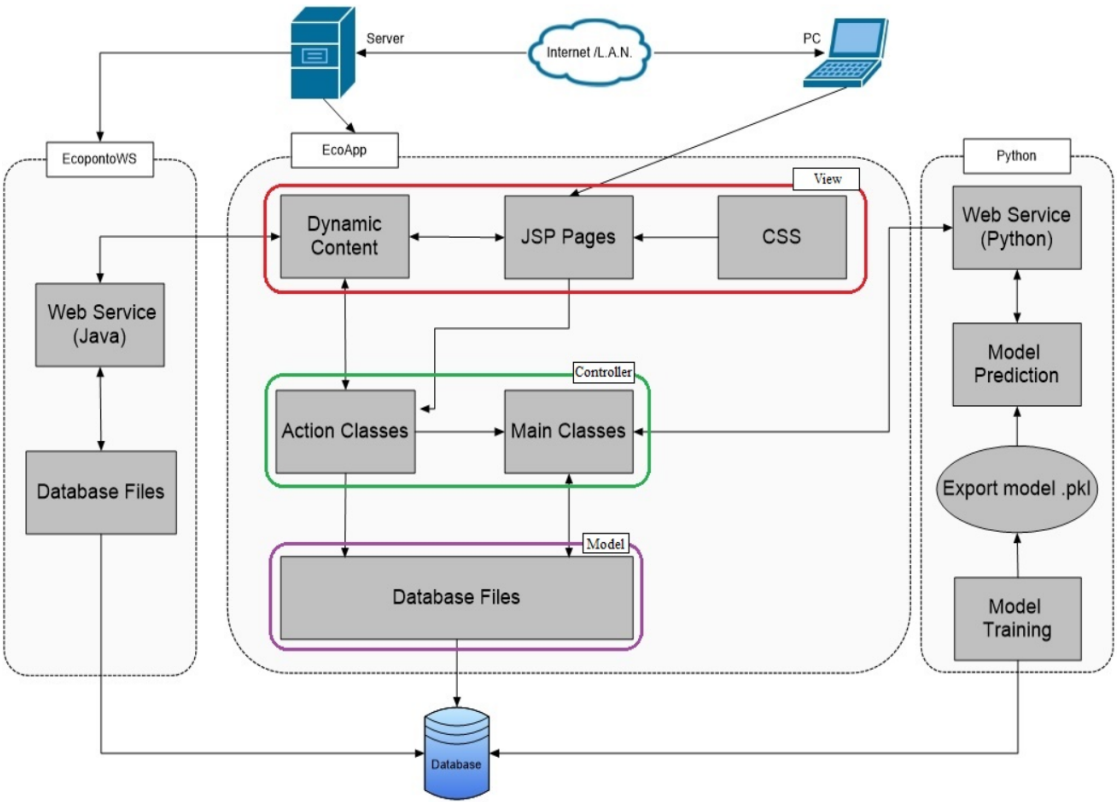


Figura 5 – Arquitetura geral

- ***View***: Está é a camada de interface com o utilizador, recebe os dados inseridos por este e apresenta o resultado das suas ações;
- ***Controller***: Esta camada controla o fluxo de informação das ações realizadas. É o intermediário entre as camadas adjacentes e controla a informação passada entre elas;
- ***Model***: Esta camada é responsável pelo armazenamento, processamento e manipulação dos dados. É responsável pelo acesso à base de dados.

Dentro de cada camada foi feita uma subdivisão em blocos, aumentado, desta forma, o nível de detalhe da arquitetura estando mais perto da especificação de implementação. Os blocos funcionais apresentados na figura anterior compõem a estrutura da aplicação.

De seguida apresenta-se a definição de cada um dos blocos, à exceção das camadas de *Web Services*, desde o seu funcionamento à interligação com os restantes elementos que constituem a camada e outras camadas da arquitetura.

- ***Dynamic Content*:**

Este bloco representa os ficheiros *JavaScript* associados à página, responsáveis pelos efeitos dinâmicos da página, permitindo uma interatividade mais alargada com o utilizador e tornando a experiência de utilização mais satisfatória. São também estes que permitem a chamadas *AJAX* que são feitas de um modo assíncrono.

- ***JSP Pages*:**

Este bloco representa as páginas *JSP* que serão visíveis pelo utilizador final e é através deste que se efetuará a sua interação com a aplicação.

- ***CSS*:**

Este bloco representa as folhas de estilo (*CSS*) que podem ser atribuídas às páginas *JSP* definindo o aspeto geral das interfaces.

- ***Action Classes*:**

Este bloco representa as classes associadas às ações sobre o sistema. Estas ações são requeridas por ações do utilizador sobre a interface ou mecanismos internos da própria aplicação.

- ***Main Classes*:**

Este bloco representa as classes principais da aplicação responsáveis por toda a gestão e processamento dos dados após uma determinada ação realizada sobre a interface. Estas classes fazem a ligação com a camada de modelo para obter o resultado do processamento dos dados lógicos.

- ***Database Files*:**

Este bloco representa as diversas classes de domínio em *Java* e os ficheiros *XML* responsáveis por identificar as entidades e associações da base de dados. Os ficheiros *XML* devem estar mapeados para o estabelecimento da ligação com a base de dados.

A aplicação encontra-se no servidor *web* como representado na arquitetura geral. O servidor é responsável por aceitar os pedidos feitos via protocolo *HTTP* dos clientes. Os utilizadores do sistema interagem com a aplicação por intermédio de uma interface *web* apresentada num navegador numa ou mais máquinas onde o canal de comunicação é a Internet ou a rede local.

Neste projeto é usado o servidor *Tomcat* [11] como servidor *web*. O *Tomcat* é um servidor *Apache* que serve a tecnologia *Java* (*JSP* e *Servlet*) permitindo o seu funcionamento em modo *web* sendo um sistema auxiliar ao *Apache*. Para este projeto, foi também abordado o *Apache HTTP Server* que é um servidor genérico que responde, não só, ao protocolo *HTTP* como outros protocolos, tais como o *HTTP*S (*HTTP* combinado com a camada de segurança *SSL*), *FTP*, entre outros. A sua utilização mais frequente é feita com a linguagem de programação *PHP* e a base de dados My*SQL*.

As funcionalidades do servidor *Tomcat* adequam-se às características tecnológicas da aplicação desenvolvida em comparação com o *Apache*. A Tabela 2 mostra os aspetos diferenciadores entre os dois servidores *web* que justificam a escolha do *Tomcat*.

Tabela 2 - Tomcat versus Apache

Tomcat	***Apache***
Fornece a funcionalidade *JSP*	Não disponibiliza a funcionalidade *JSP*
Tempos de carregamento e reimplementação mais rápidos com apenas uma funcionalidade a ser executada no servidor	Oferece mais funcionalidades para execução aumentado o tempo de carregamento e reimplementação do servidor
Especificamente criado para *Java*	Pode usar diferentes linguagens de programação (*PHP*, *Python*, *Perl*)
Pode adicionar uma camada extra de segurança ao servidor	Vulnerabilidades de segurança
Desempenho estável para sites com tráfego muito alto por ser um servidor leve (funcionalidade básica)	Problemas de desempenho em *websites* com tráfego muito alto

Para aplicações totalmente construídas na tecnologia *JSP*, o *Tomcat* é a melhor opção. Este servidor permite um ambiente *Java* reunindo as tecnologias baseadas nesta linguagem de programação.

Outro fator relevante é a organização dos ficheiros no servidor, é importante que a organização seja um paralelo com a arquitetura lógica da aplicação dividindo os ficheiros por camadas facilitando o modo de acesso e não misturar os conteúdos de cada camada para não dificultar o desenvolvimento da aplicação.

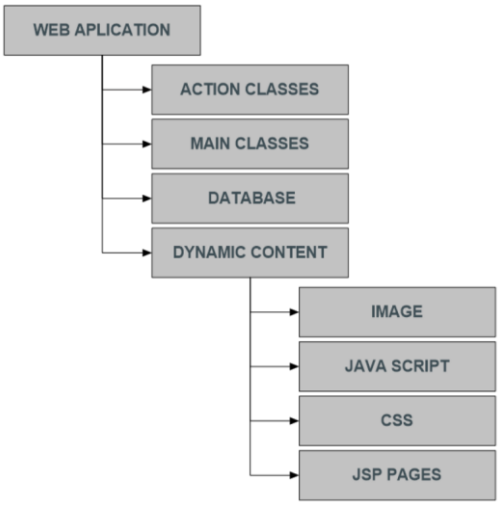


Figura 6 - Organização dos ficheiros no servidor

Ao organizar a aplicação em pastas contendo os ficheiros faz com que o controlo da organização e do acesso ao conteúdo do servidor se torne mais simplificado.

O conteúdo de multimédia começou por estar organizado no sistema de ficheiros do servidor como apresentado na Figura 6. A partir do momento que a aplicação passou a ser executada na Internet e não via local, foi necessário recorrer a um serviço externo para organizar o conteúdo multimédia da aplicação e, para esse efeito foi usado o *Amazon S3.*

Este serviço é fornecido pela *Amazon Web Services* (AWS) que fornece o armazenamento de objetos por meio de uma interface de serviço da *web*. O *Amazon S3* permite a organização por pastas ("*buckets*") com os ficheiros da aplicação a serem introduzidos nelas.

3.1.2 Enquadramento Tecnológico

As escolhas tecnológicas tiveram como base não só as necessidades do projeto, mas também a experiência passada adquirida ao longo do curso onde muitas das tecnologias abordadas neste projeto são lecionadas e detalhadas permitindo o conhecimento necessário para o desenvolvimento do mesmo. Pretende-se manter uma certa uniformidade nas tecnologias utilizadas neste projeto por razões de continuidade e manutenção. As tecnologias são definidas como tecnologias *front-end* (cliente) ou *back-end* (servidor).

HTML

HTML5 é uma linguagem baseada em marca ("*markup language*") utilizada na construção de páginas *web* de modo que possam a ser interpretados pelos *browsers*. A versão 5 do *HTML* (*HTML5*) traz novas e importantes mudanças em relação à sua anterior versão. Foram acrescentadas (ao *HTML5*) novas funções que simplificam a inclusão e a manipulação de conteúdo gráfico e multimédia na *web* sem ter de recorrer a plugins ou API. O código é embebido dentro das páginas *JSP* criadas na aplicação. Esta tecnologia é usada no *front-end* da aplicação.

CSS

CSS é uma linguagem de estilos utilizada para definir a apresentação de documentos em linguagens de marcação, como o *HTML* ou *XML*. Uma das vantagens de usar a linguagem *CSS* é o fato de o tamanho do ficheiro *CSS* ser reduzido e com isto ajudar ao carregamento de páginas *web* de forma mais rápida, e permitindo que as alterações feitas possam ser aplicadas a todo o documento. É uma tecnologia *front-end* usada nas páginas *JSP* da aplicação.

JavaScript

É uma linguagem de script para programação maioritariamente no *front-end* de uma aplicação *web*. O principal uso desta linguagem é escrever funções para serem incluídas em páginas *HTML*, esta linguagem é definida com o uso das *tags*. O *JavaScript* utiliza os seus *scripts* para interagir com o utilizador sem ter de passar pelo *back-end* realizando comunicação assíncrona e alterando o conteúdo da página exibida. O maior objetivo do *JavaScript* é inserir conteúdo dinâmico nas páginas *web*.

JSP

A *Java Server Page* (*JSP*) é uma tecnologia que tem como objetivo criar páginas dinâmicas baseadas em *HTML*. Estas páginas permitem o uso do *Java* intercalado com o conteúdo do código *web*, com a página a ser compilada e executada no servidor. Nesta aplicação, a *JSP* é usada como componente de vista fazendo a interação com os utilizadores no *front-end*.

Java

O *Java* é uma linguagem de programação orientada a objetos, projetada para permitir o desenvolvimento de aplicações para a mais ampla variedade possível de plataformas [12]. É uma linguagem usada no *back-end* de uma aplicação onde é feito o processamento lógico dos dados do servidor. Dentro das páginas *JSP* é usado código *Java*.

Para efetuar a ligação e chamadas à base de dados é utilizado o código desta linguagem para submeter os pedidos ao sistema de gestão de base de dados.

PostgreSQL

O *PostgreSQL* é um dos sistemas de gestão de base de dados mais avançados com muitos recursos disponíveis o que foi um fator importante na escolha de utilização para este projeto. É uma base de dados altamente escalável, tanto na grande quantidade de dados que pode guardar e gerir como no número de utilizadores em simultâneo, assim como o *MySQL*.

Python

O *Python* é uma linguagem de programação de alto nível, dinâmica, com extensas bibliotecas padrão e módulos para quase todas as necessidades. Pode permitir extensas funcionalidades como o suporte às técnicas de *deep learning* devido às suas bibliotecas especializadas nestas técnicas. É uma linguagem de programação usada no *back-end.*

Outros Aspetos de Tecnologias

Para além das tecnologias já mencionadas, e com o objetivo de disponibilizar uma interface melhor ao utilizador foram incluídas as tecnologias de *AJAX*, *JQuery* e *JSON*.

O *AJAX* é uma tecnologia que numa determinada página *HTML* pode fazer chamadas de forma assíncrona para o servidor e carregar o conteúdo deste, que pode ser em formato *XML*, *HTML* ou objetos *JavaScript*. É um método usado de modo que uma determinada página *web* seja alterada recebendo ou não informação do servidor *web*, em que seja necessário fazer *reload* à página questão [13]. Alguns exemplos da interação do *AJAX*.

- Validação de dados em tempo real;
- Carregar informação com base de um evento do utilizador;
- Atualizar dados e comunicar com o servidor.

O *JSON* [14] é um formato para a troca de dados computacionais entre sistemas independentemente da linguagem de programação derivado do *JavaScript*. É tipicamente usado em ambientes com grande fluxo de dados entre o cliente e o servidor. Os *Web Services* e a tecnologia *AJAX* usam de forma frequente o *JSON*.

O *JQuery* [15] é uma biblioteca de funções *JavaScript* que interage com o *HTML*. Foi desenvolvida para simplificar os scripts interpretados no navegador *front-end* capaz de simplificar a manipulação dos vários elementos de uma página *web*. Com o *JQuery* é rápido e fácil de enriquecer qualquer aplicação *web* com efeitos, estilos, animação e a utilização do *AJAX*.

Estas tecnologias permitem o aumento do desempenho da aplicação, a redução de código e uma melhor gestão de processamento.

Ferramentas

É possível utilizar *frameworks* para facilitar o desenvolvimento de aplicações *web*, e por isso, foram utilizadas as seguintes neste projeto:

- ***Struts2*** [16]**:**

É uma ferramenta utilizada para o desenvolvimento de aplicações *web* em *Java* usando e estendendo a API *Java Servlet* facilitando a integração com o *Java* a nível de código. Todas as ações que são feitas pelo cliente são enviadas para o *Struts2* que faz o controlo destas ações e as envia para a camada de modelo para serem processados os dados. O *Struts2* encontra-se sempre na camada de controlo e usa código *Java* para realizar o processamento das ações que são definidas na ferramenta.

- ***Hibernate*** [17]**:**

A ligação entre o *Java* e base de dados é feita pela ferramenta *Hibernate*. Esta ferramenta é usada para o mapeamento objeto-relacional escrito na linguagem *Java* facilitando o mapeamento dos atributos entre o modelo relacional e o modelo objeto de uma aplicação mediante o uso de ficheiros *XML* e classes em *Java*.

A principal característica do *Hibernate* é a transformação entre classes *Java* e tabela de dados. Estas classes representam os dados das tabelas que se encontram na base de dados. O *Hibernate* gera as chamadas *SQL* para qualquer base de dados.

- ***Flask*** [18]**:**

É uma ferramenta *web* (código escrito em *Python*) usada para o desenvolvimento de aplicações *web* e *Web Services*.

- ***Bootstrap*** [19]**:**

Para a implementação das páginas visualizadas pelo utilizador foi usado o *Bootstrap* que contém código *HTML*, *CSS* e *JavaScript* para desenvolver projetos *responsive*. Esta ferramenta permite a facilidade em desenvolver código fazendo chamadas das classes que estão definidas nas tecnologias do *Bootstrap*.

- ***Gunicorn*** [20]**:**

É um servidor *HTTP* da interface *gateway* do servidor *web Python*.

Open-Source

Uma ferramenta *open-source* é aquela que tem o seu código aberto, ou seja, pode ser visualizada por qualquer pessoa sendo disponibilizada a versão desenvolvida pelo programador ao público, sem restrições. O código-fonte de um programa diz respeito à sua estrutura escrita numa linguagem de programação que permite ser executado por um computador [21].

As ferramentas de *open-source* oferecem uma série de vantagens tanto para programadores como para utilizadores que apostem nessas soluções.

- **Personalização:** As ferramentas *open-source* fornecem o seu código-fonte não apenas para leitura, mas também permitindo a sua adaptação. Dessa forma, os utilizadores que tenham conhecimento na linguagem de programação adotada pelo programa podem fazer adaptações para melhorar a experiência da solução;
- **Controlo:** Ao contrário dos softwares prioritários, as ferramentas *open-source* podem ser corrigidas a qualquer momento e por qualquer programador, sem precisar de esperar por atualizações da empresa criadora da solução;
- **Menos gastos:** Este é dos pontos mais populares sobre o conceito de *open-source*, já que, não é necessário pagar por uma licença de uso do software;
- **Transparência:** A privacidade é uma das maiores preocupações no desenvolvimento de um software. Quando o código é fechado não dá para ter a certeza dos dados guardados. Nas ferramentas *open-source*, o programador tem acesso a toda a sua estrutura;
- **Interação:** As ferramentas *open-source* adaptam-se com mais facilidade a softwares de origens variadas.

Apesar das vantagens mencionadas, a aposta em ferramentas *open-source* tem as suas desvantagens, conforme apresentamos em seguida:

- Sem garantias de proteção de dados ou entidades que fiscalizem todas as soluções com código aberto;
- Menor segurança devido à livre configuração, o que pode gerar problemas para plataformas de lojas virtuais, por exemplo, expondo os dados financeiros dos clientes;
- Exige conhecimentos avançados em programação, incluindo o domínio da linguagem utilizada para escrever o código-fonte do programa;
- Dependendo do programa, pode resultar em plataformas com carregamento lento e restrições quanto ao *layout*.

Neste projeto, são usadas ferramentas *open-source* o que facilita o desenvolvimento de alterações à aplicação atual adaptando novas tecnologias como trabalho futuro.

3.1.3 *Web Services*

Um *Web Service* é um método disponível para invocação por outros programas utilizando tecnologias *web*. É muitas vezes usado para transferir dados através de protocolos de comunicação para diferentes plataformas.

A utilização de *Web Services* traz vários benefícios a nível tecnológico, nomeadamente:

- **Integração de informação e sistemas:** uma vez que o funcionamento do *Web Service* necessita apenas da tecnologia *XML*/*JSON* e protocolos *HTTP*, a comunicação entre sistemas e aplicações é bastante simplificada, sendo possível a troca de informações entre dois sistemas;
- **Reutilização de código:** um *Web Service* pode ser utilizado por várias plataformas com diferentes objetivos. O código é feito uma vez e pode ser utilizado vezes sem conta por diferentes aplicações;
- **Maior segurança:** o *Web Service* evita que se comunique diretamente com a base de dados salvaguardando os dados inseridos na base de dados.

Para garantir a comunicação entre o *Web Service* e o sistema que faz o pedido é necessário uma linguagem intermédia. Para tal, existem protocolos de comunicação como o *SOAP* e o *REST*.

O protocolo *SOAP* utiliza *XML* para enviar mensagens e, geralmente, serve-se do protocolo *HTTP*. O *SOAP* tem associado o documento *WSDL* que descreve a localização do *Web Service* e as operações disponíveis.

O *REST* é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de *Web Services* e surgiu com o objetivo de simplificar os acessos a eles baseando-se no protocolo *HTTP* e permitindo utilizar vários formatos para a representação de dados como o *JSON*, *XML*, *RSS*, entre outros. Os *Web Services* que estão em conformidade com o estilo da arquitetura *REST* são denominados de *RESTFul*.

A Tabela 3 mostra as diferenças entre ambos em determinadas características.

Tabela 3 - SOAP versus REST

Características	***SOAP***	***REST***
Tipo	Protocolo	Arquitetura
Função	Transferir informação estruturada	Aceder a um recurso para dados
Formato de Dados	*XML*	*XML*, *HTML*, *RSS* e *JSON*
Segurança	Suporta *WS-Security* e *SSL*	Suporta *SSL* e *HTTP*S
Largura de Banda	Requer mais recursos e largura de banda	Requer poucos recursos e pouco uso de largura de banda

Comparando os dois serviços observamos que o *REST* é mais fácil e vantajoso de usar em relação ao *SOAP* no contexto do projeto. O *REST* permite a comunicação em vários formatos de dados o que facilita a escolha das tecnologias a usar, não estando dependente só de um formato limitando assim a nível tecnológico a aplicação. O *REST* não usa muitos recursos o que não sobrecarrega a aplicação em termos de desempenho, visto que a perspetiva é ser usada por muitos utilizadores.

As grandes empresas como a *Google*, *Facebook*, *Amazon*, *Microsoft*, entre outros começaram a usar serviços *REST* e tem crescido amplamente a sua aderência e suporte da tecnologia por parte destas empresas. Foi com este intuito e motivos que foram utilizados os *Web Services RESTFul* na aplicação desenvolvida neste projeto.

A arquitetura geral do projeto apresenta duas camadas de *Web Services*, uma executada no servidor (código *Java*) e outra que comunica com o servidor (código *Python*). O modo de funcionamento de cada um vai ser apresentado nas próximas figuras.

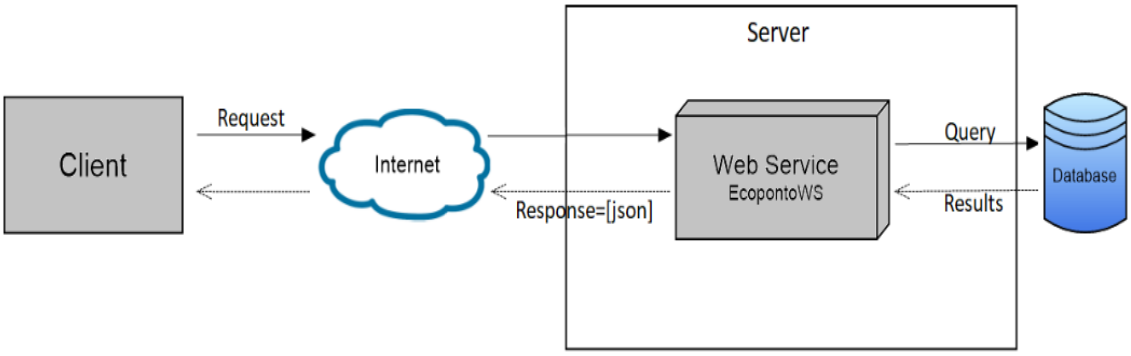


Figura 7 - Arquitetura RESTFul (Java)

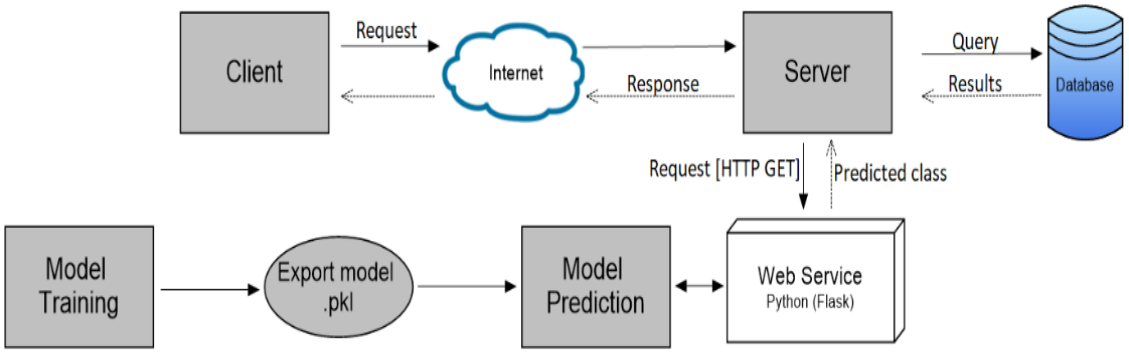


Figura 8 - Arquitetura RESTFul (Python)

3.1.4 Sistema de Geolocalização

A Geolocalização é um processo de localização de um determinado objeto através da atribuição de coordenadas geográficas. Estas coordenadas podem ser obtidas através de várias formas:

- **Wi-Fi:** a distância é determinada de acordo com a intensidade do sinal, quanto mais longe, menos intenso e vice-versa;
- **GPS:** três, ou quatro satélites, presentes na órbita da Terra, fornecem as coordenadas (latitude e longitude) de um ponto do nosso planeta;
- **Radiofrequência:** como o nome sugere, essa forma de geolocalização é pautada na emissão de ondas de rádio;
- **AGPS:** também conhecido como GPS assistido. Esta é a evolução do GPS, além dos satélites também são usadas antenas de telemóveis na geolocalização.

Referente ao processo de geolocalização existem várias vantagens sobre a sua utilização, nomeadamente:

- Localização em tempo real;
- Fornecimento atualizado de informações;
- Deteção de locais específicos para onde se pretende ir;
- Planeamento do caminho para um local específico.

No desenvolvimento de aplicações *web* é possível integrar ferramentas de geolocalização sendo um dos maiores exemplos a API do *Google Maps* [22].

O *Google Maps* consiste num serviço de pesquisa e visualização de mapas e imagens de satélite da Terra fornecido e desenvolvido pela *Google*. O serviço disponibiliza mapas e rotas para qualquer ponto do mundo, para além disso disponibiliza, também, imagens de satélite do mundo com a possibilidade de fazer zoom sobre muitas das cidades. Com a conta da API da *Google* é possível destacar as próprias rotas, pontos e áreas e outras funcionalidades.

Neste projeto, foi integrada a API do *Google Maps* onde as suas funcionalidades são disponibilizadas e invocadas através do *Web Service RESTFul* criado em *Java*.

3.1.5 Servidor de Email

O *SMTP* é um protocolo de transferência de email que especifica a informação que identifica cada email e o caminho que ele deve percorrer para ser entregue de forma segura. Entre as diferentes etapas do caminho, a passagem por um servidor *SMTP* é uma das mais relevantes.

Existem servidores gratuitos que podem ser usados em contexto de desenvolvimento, nomeadamente:

- *Gmail;*
- *Hotmail;*
- *Yahoo;*
- *Outlook.com.*

Tabela 4 - Servidores SMTP

Email	Servidor *SMTP*	Porta
Gmail	*SMTP.gmail.com*	465 para SSL e 587 para *TLS*
Hotmail	*SMTP.live.com*	25 ou 465
Yahoo	*SMTP.mail.yahoo.com*	465
Outlook.com	*SMTP.office365.com*	587

Atualmente, existem API disponíveis que fornecem o serviço *SMTP* permitindo aos programadores integrar na sua aplicação as configurações e parâmetros destes serviços fornecendo os recursos necessários para serem usados.

O envio de emails é feito, apenas com o nome do servidor *SMTP*, porta, nome de utilizador e *password*. Estes parâmetros são configurados na API e usados na aplicação para permitir a comunicação entre ambos.

3.1.6 Criptografia

A segurança de uma aplicação é bastante importante para evitar ataques que possam afetar os *web*sites. Neste projeto é relevante que este seja um dos princípios base da estrutura da aplicação.

A criptografia é uma área que se refere à construção e analise de protocolos que impedem terceiros, ou público, de acederem a mensagens privadas. Muitos aspetos em segurança de informação como a confidencialidade, integridade de dados e autenticação são centrais à criptografia, atualmente. Aplicações de criptografia incluem o comércio eletrónico, cartões de pagamento baseados em *chip*, moedas digitais, *passwords* de computadores e comunicações militares [23].

Uma informação não cifrada que é enviada de uma pessoa para outra é chamada de "texto em claro". A cifragem é o processo de conversão de um texto em claro para um código cifrado e a decifragem faz o processo inverso, de recuperar o texto original a partir de um texto cifrado.

Existem vários métodos e algoritmos de uso da criptografia como a criptografia de chave simétrica que pode usar cifras de fluxo, cifras de bloco ou funções em *hash*, neste caso, foi usado o algoritmo de criptografia MD5.

Ele foi projetado, inicialmente, para ser usado como uma função *hash* criptográfica e, ainda, pode ser usado como uma soma de verificação para analisar a integridade de dados.

Segundo, a *CMU Software Engineering Institute*[2] o MD5 é considerado essencialmente uma criptografia inadequada visto que tem alguns pontos fracos, como falsificar uma assinatura digital. Apesar de algumas fraquezas, este algoritmo tem algumas vantagens, tais como:

- Ser mais fácil de comparar e armazenar *hash* menores do que armazenar um texto grande;
- Usado para *hash* unilaterais que são usados sem fornecer o valor original;
- Ter facilidade em gerar um resumo da mensagem original usando este algoritmo.

O funcionamento do algoritmo de mensagem MD5 consiste no processamento de dados em blocos de 512 bits, divididos em 16 palavras compostas por 32 bits cada. O resultado do MD5 é um valor de mensagem de 128 bits.

O cálculo do valor do MD5 é executado em várias etapas que processam cada bloco de dados de 512 bits junto com o valor calculado na etapa anterior. A primeira etapa começa com os valores iniciais da mensagem de MD5 usando valores numéricos hexadecimais consecutivos. Cada etapa inclui quatro mensagens MD5 que manipulam valores no bloco de dados atual e processados no bloco anterior. O valor final calculado a partir do último bloco torna-se a mensagem MD5 para esse bloco [24].

Neste projeto, foi implementado o algoritmo MD5 usado para permitir a segurança da aplicação no sistema de autenticação dos utilizadores, cifrando as *passwords* no momento do registo.

---

[2] **CMU Software Engineering Institute,** é um centro de pesquisa e desenvolvimento relacionado com a área de Engenharia de Software patrocinado pelo Departamento de Defesa dos Estados Unidos [25].

3.1.7 Redes Neuronais

Uma rede neuronal consiste numa estrutura de ligações, na qual o processamento está distribuído por um grande número de pequenas unidades densamente interligadas. São modelos computacionais inspirados pelo sistema nervoso central que são capazes de realizar a aprendizagem automática supervisionada.

Assim, como outros métodos de aprendizagem automática e sistemas que aprendem a partir de dados, as redes neuronais têm sido usadas para resolver uma grande variedade de tarefas que são difíceis de resolver utilizando programação baseada em regras comuns como a visão computacional e o reconhecimento de voz.

O objetivo da aprendizagem é o de minimizar uma função de erro (a rede neuronal resolve assim um problema de otimização).

Uma das possibilidades mais explorada das redes neuronais incide na classificação de padrões de entradas em diferentes categorias. Desde o reconhecimento de diversos tipos de terreno em imagens de satélite à identificação de veículos em imagens podemos fazer a classificação através das redes neuronais.

Existem diversos tipos de redes neuronais, mas especificamos neste projeto a rede neuronal utilizada em aprendizagem profunda (*deep learning*). O *deep learning* consiste num ramo da aprendizagem automática baseado num conjunto de algoritmos com várias camadas de processamento, compostas por várias transformações lineares e não lineares.

Na classificação de imagens, o *deep learning* usa a *convolutional neural network* (*CNN*), que é considerada uma classe de redes neuronais de aprendizagem profunda. As *CNN* representam um grande avanço no reconhecimento de imagem. Elas são usadas, normalmente, para analisar imagens e depois classificá-las.

A classificação de imagem consiste em obter a probabilidade de a imagem pertencer a determinada classe. Uma *CNN* realiza a convolução de matrizes que representam operadores que extraem características ("*features*") das imagens e otimiza esses operadores de modo a minimizar o erro global da rede, ou seja, a *CNN* aprende a extrair as "*features*" que permitem distinguir diferentes imagens. Isto significa que este tipo de rede é ideal para o processamento de imagem em 2D. A próxima figura apresenta a típica arquitetura da *CNN*.

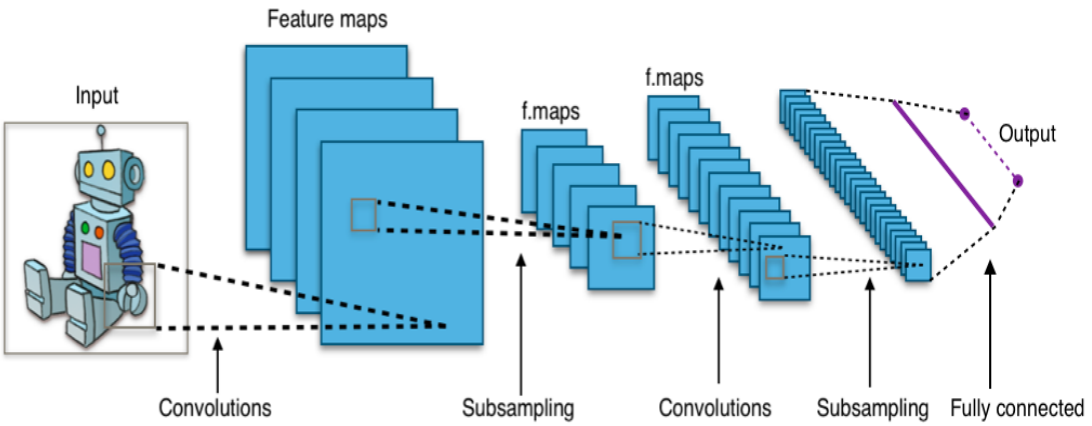


Figura 9 - Arquitetura CNN [26]

Atualmente, é possível fazer a classificação de imagens de um *dataset* específico com base em arquiteturas *CNN* de referência, o que permite que o programador tenha o trabalho facilitado. Neste caso, foi usada a rede neuronal residual *ResNet* [26]. A *ResNet* é um tipo de rede neuronal que aplica o mapeamento de identidade, o que significa que a entrada para uma camada passa diretamente ou passa como atalho para outra camada. Na Figura 10, é representado o esquema lógico da base de construção de blocos das redes *ResNet*.

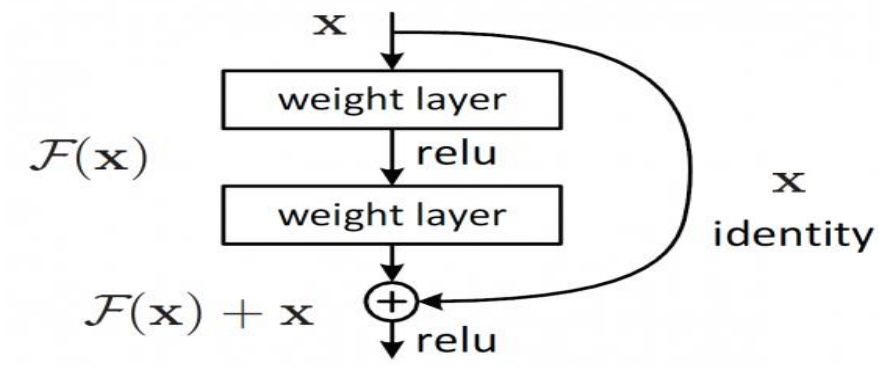


Figura 10 - Esquema lógico ResNet [27]

Pela figura acima, podemos verificar o conceito mais importante envolvido que é a ligação por atalho. O atalho é o mapeamento de identidade, em que a entrada da camada anterior é adicionada diretamente à saída da outra camada.

Em termos simples, uma *ResNet* possui dois tipos de blocos residuais, o primeiro é composto por duas ou mais convoluções cuja saída final possui o mesmo tamanho do que a entrada. Após passar pelas duas camadas do bloco o resultado é somado, elemento a elemento à entrada do bloco.

Por exemplo, suponhamos que a entrada X do bloco seja uma imagem colorida de 128x128x3, a saída após a segunda camada também deve possuir as mesmas dimensões da entrada somando no fim a entrada X, sendo obtida a saída do bloco residual.

Cada bloco de *ResNet* tem duas camadas de profundidade (usado em redes *ResNet* 18 e 34) ou três camadas de profundidade (*ResNet* 50, 101 e 152).

De acordo com alguns estudos e avaliações feitas [27], o *ResNet*34 contém melhores resultados do que o *ResNet18*. A rede neuronal residual converge mais rápido em comparação com as redes "*plain*". As redes "*plain*" são consideradas redes não residuais, pois não usam o esquema lógico das *ResNet*. Todos os blocos estão interligados sem o uso de qualquer atalho, aumentando assim, a degradação do gradiente. As próximas figuras mostram os resultados do conjunto de imagens de treino entre as duas redes.

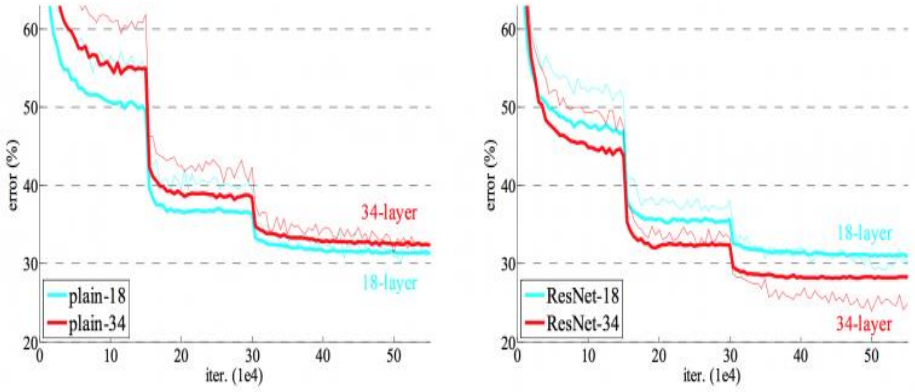


Figura 11 - Resultados entre redes "plain" e ResNet. Curva com taxa de erro [27]

Entre as redes *ResNet*, a rede com mais camadas consegue ter melhores resultados em relação às outras com menos camadas. A taxa de erro é muito menor à medida que os ciclos de treino vão aumentado. A *ResNet34* atinge um erro de 5.71%, enquanto a *ResNet*152, que tem o melhor resultado, atinge 4.49%.

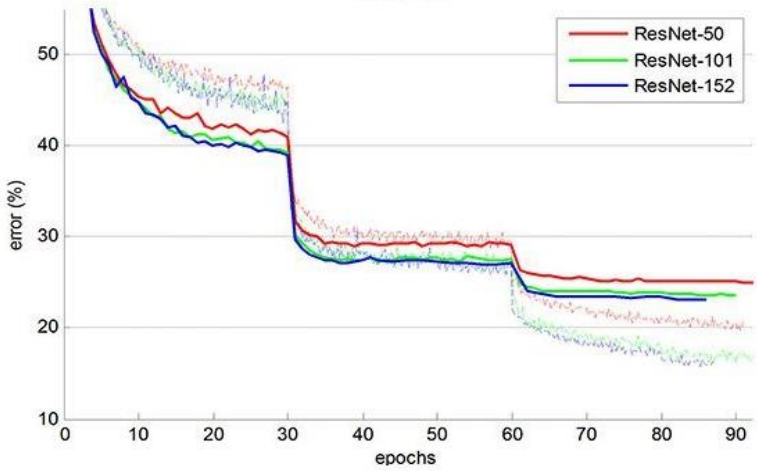


Figura 12 - Taxa de erro de redes ResNet com aumento dos ciclos [27]

Neste caso, o *Resnet34* é usado para este projeto por ter melhores resultados em relação à *ResNet18* e às redes "*plain*" levando menos tempo a treinar, usando menos memória e obtendo uma taxa de erro aceitável para o desejado.

As redes de três camadas de profundidade obtêm melhores resultados, mas levam mais tempo usando mais memória a treinar o que para o objetivo deste projeto não é o pretendido.

Usando estas *ResNet*, vários problemas podem ser resolvidos, tais como:

- Menor taxa de erro com o aumento da profundidade de rede do que em relação a outras redes;
- Podem facilmente obter a precisão com uma profundidade muito maior produzindo resultados muitos melhores.

3.2 Requisitos

Os requisitos descrevem e sistematizam as características que as aplicações devem exibir para satisfazer as necessidades dos utilizadores. Em geral, o conceito de requisito é a característica ou a capacidade da aplicação necessária à concretização dos objetivos para a qual é concebida. Existem dois tipos de requisitos funcionais e não funcionais:

Os requisitos funcionais:

- Descrevem a interação entre a aplicação e o seu ambiente;
- Descrevem como a aplicação se deve comportar dada uma determinada ação.

Os requisitos não funcionais:

- Descrevem as restrições que a aplicação deve satisfazer ao responder aos requisitos funcionais.

Após a identificação dos requisitos, é necessária a sua especificação através dos modelos de caso de utilização que representam os requisitos funcionais elaborados na perspetiva dos utilizadores. A especificação suplementar representa os requisitos não funcionais.

3.2.1 Requisitos Funcionais

Os requisitos funcionais da aplicação desenvolvida estão organizados em vários subsistemas, que são os seguintes:

- Gestão de Utilizadores (RF1);
- Gestão de Acessos (RF2);
- Perfil de Administrador (RF3);
- Gestão Ecopontos, *Links*, Eventos e Imagens/Vídeos (RF4);
- Perfil de Utilizador (RF5);

- Processamento de classificação automática de imagens de resíduos (RF6).

Seguidamente, apresenta-se, como exemplo, os requisitos funcionais relacionados com a Gestão de Utilizadores na aplicação. Os restantes requisitos funcionais estão identificados em anexo (cf., Anexo B).

Tabela 5 - Requisitos funcionais da gestão de utilizadores

ID Requisito Funcional	Descrição Requisito Funcional
RF1.1	Novos registos de utilizadores na aplicação.
RF1.2	Alteração da *password* do utilizador.
RF1.3	Alteração de dados do utilizador.
RF1.4	Remover utilizadores da aplicação.

A aplicação desenvolvida deve suportar os requisitos típicos da gestão de utilizadores e é fundamental que seja necessário fazer a gestão de acessos deles mesmos. Existem dois tipos de perfis que são: o de Utilizador e o de Administrador.

3.2.2 Requisitos não Funcionais

Em seguida, apresenta-se os requisitos não funcionais da aplicação.

- A aplicação *web* deve ter um sistema de segurança de *login* fazendo a cifragem da *password*;
- A aplicação deverá ser executada em qualquer *browser*;
- A aplicação deverá comunicar e registar os dados de forma persistente na base de dados *PostgreSQL;*
- A aplicação deverá sempre validar o período de inatividade de uso da aplicação por parte do utilizador ao iniciar cada sessão;
- A aplicação deve ter um *time-out* de fecho de sessão após inatividade;
- A aplicação deverá comunicar entre a linguagem de programação *Java* e *Python* para o processo de reconhecimento de imagens;
- A aplicação *web* deverá ser alojada num domínio na Internet.

Estes são os requisitos não funcionais da aplicação, em que são abordadas várias características como a segurança, usabilidade, portabilidade, implementação, interoperabilidade e a eficiência da aplicação.

3.3 Casos de Utilização

Os casos de utilização descrevem os requisitos funcionais da aplicação e a interação entre os vários atores e a aplicação. Estes casos contêm os objetivos que os atores têm ao usar a aplicação. É fundamental que a identificação de requisitos tenha sido correta para poder descrever a sequência de ações de um utilizador nos casos de utilização.

Tendo em conta que se pretende criar uma aplicação *web* foram identificados os atores e algumas das ações que são realizadas. Procedeu-se à exposição dos casos de utilização recorrendo ao Diagrama de Casos de Utilização.

O Diagrama de Casos de Utilização da figura seguinte representa os atores do sistema e algumas das suas possíveis ações.

Figura 13 - Diagrama de casos utilização do sistema

Um caso de utilização descreve o que faz uma aplicação (ou parte desta), mas não como é que tal é realizado. O foco é na visão externa da aplicação, ou seja, na visão que os utilizadores têm dele.

Para cada caso de utilização identificado, foi elaborada a respetiva especificação onde são identificados os atores presentes e todos os passos principais e alternativos até ao fim do caso de utilização. Devido ao grande número de casos de utilização, apenas será apresentado na Tabela 6, a título de exemplo, a descrição de um caso de utilização, podendo todos os outros ser consultados em anexo (cf., Anexo B). O caso de utilização escolhido para a especificação é o de "*Upload Imagem para Reconhecimento*" que corresponde ao processo de classificação da imagem.

Tabela 6 - Caso de utilização "Upload Imagem para Reconhecimento"

Nome: *Upload* Imagem para Reconhecimento **Descrição**: Permite ao Utilizador identificar o tipo de material da imagem que carrega na aplicação **Atores**: Utilizador
Cenário principal: 1. O Utilizador encontra-se na página de *homepage* da aplicação w*eb* após o *login* 2. O Utilizador clica sobre o seu nome na página e carrega em "Perfil". 3. Na página de perfil clica em "Adicionar" e faz *upload* da imagem. 4. A aplicação valida a imagem e faz o seu processo de reconhecimento. 5. A aplicação mostra o resultado da classificação e o utilizador valida a classe (corrigindo ou não a classificação do resíduo) e o caso de utilização termina. **Cenário alternativo 1:** 1. No passo 3 a imagem é inválida. 2. A aplicação mostra uma mensagem a informar o utilizador desse facto e o caso de utilização termina.

Na figura seguinte, é mostrado o diagrama de sequência com as classes pertencentes ao caso de utilização anterior.

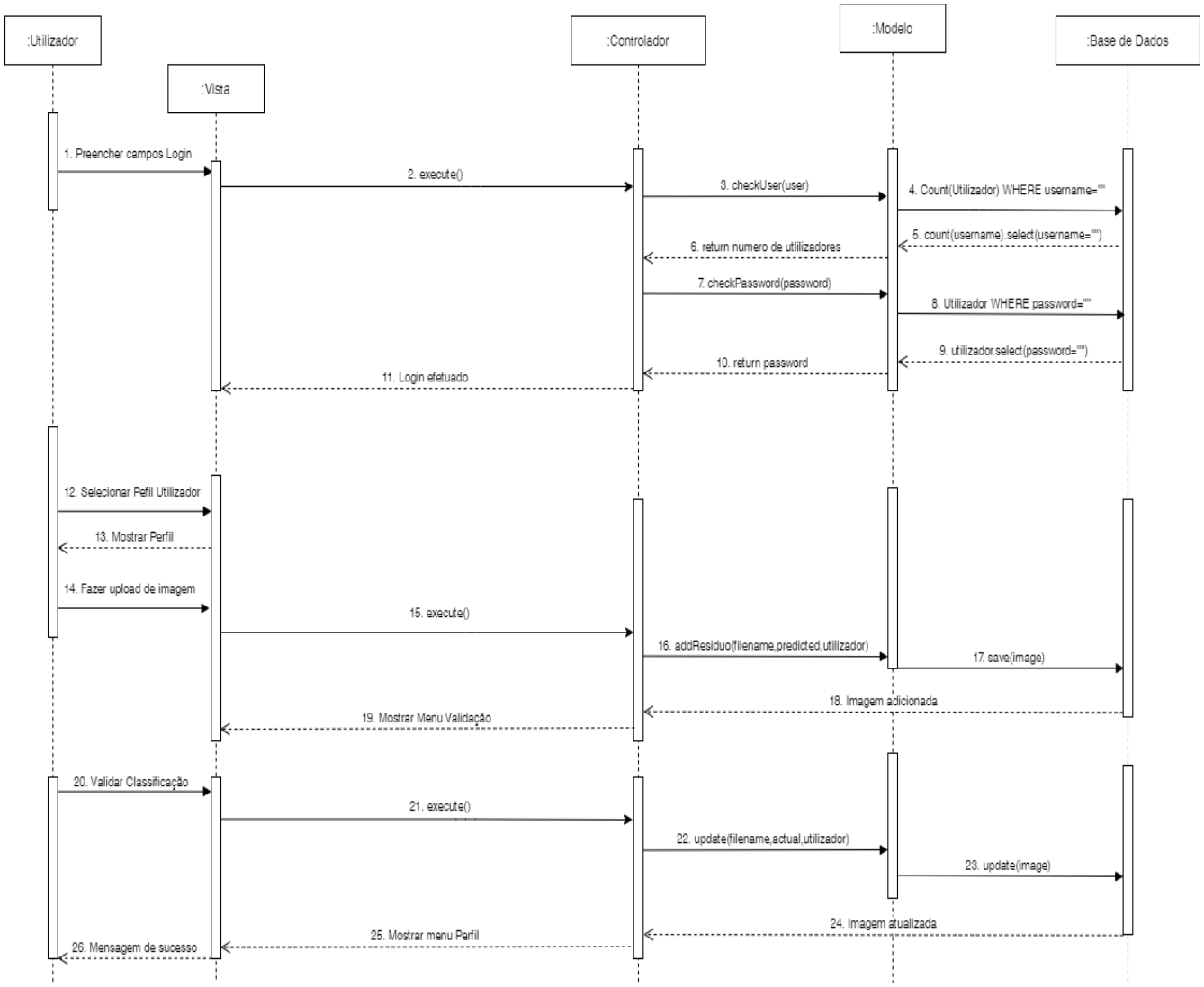


Figura 14 - Diagrama do caso de utilização "Upload Imagem para Reconhecimento"

Os diagramas de sequência dos casos de utilização da aplicação utilizam o padrão *MVC*, disponível em anexo (cf., Anexo A), construído com base nas classes:

- **Utilizador:** representa o ator (entidade externa ao sistema e que dele depende) que está a fazer as ações na aplicação. A classe Administrador aparece quando são os casos de utilização referentes ao Administrador da aplicação;
- **Vista:** representa a classe que permite ao ator fazer as suas ações. É a classe que permite a interação com o ator;
- **Controlador:** representa a classe que controla as ações do utilizador e as processa para a camada de modelo;
- **Modelo:** representa a classe que processa os dados enviados pela classe controlador;
- **Base de Dados:** armazena os dados da *EcoApp*.

3.4 Abordagem

Em virtude do que foi mencionado neste capítulo na seção 3.1, são indicados os métodos, formulações e especificações centrais do trabalho que têm um importante contributo ao nível de qualidade e enriquecimento do projeto.

3.4.1 Configuração e execução do servidor de Email

A utilização do servidor *SMTP* tem como objetivo fornecer a funcionalidade que permita o envio de emails para o utilizador em caso de esquecimento da *password* e para avisar os administradores de novos registos na aplicação.

A configuração do *SMTP* é feita numa API de email denominada *Mailgun* [28] que depois disponibiliza os parâmetros necessários à comunicação para serem usados no método da aplicação. Este método facilita a simplificação da recuperação da *password* onde através da inserção do email, é relativamente fácil a geração de uma nova *password*.

No código seguinte são apresentadas as principais configurações do servidor *SMTP*.

Código 1 - Configuração servidor SMTP

```
Properties props = System.getProperties();
props.put("mail.SMTPs.host", host);
props.put("mail.SMTP.user", user);
props.put("mail.SMTP.password", pass);
props.put("mail.SMTP.startTLS.enable", enabled);
props.put("mail.SMTPs.auth", auth);
props.put("mail.SMTP.port", port);
```

3.4.2 Implementação do algoritmo de cifragem

A cifragem a partir do algoritmo MD5 é feita com o intuito de proteger as *passwords* que são introduzidas pelos utilizadores no momento do registo na aplicação e, consequentemente, registadas na base de dados. As *passwords* são cifradas uma vez e não são decifradas em nenhum momento ficando assim protegidas de qualquer falha de segurança. O método de cifragem das *passwords* é disponibilizado em anexo (cf., Anexo B).

3.4.3 Geolocalização com API *Google Maps*

A Geolocalização disponibilizada pela API da *Google Maps* permite que o *Web Service RESTFul* consiga invocar as funções que esta API disponibiliza. As várias funcionalidades aplicadas na interface do mapa são dependentes das funções que são apresentadas na Tabela 7 onde são definidas as funções principais utilizadas e a sua finalidade.

Tabela 7 - Funções da API Google Maps

Funções	Finalidade
DirectionsService.route	Estima e mostra o caminho entre dois pontos no mapa.
Google.Maps.Marker	Especifica um ponto no mapa através de um marcador definido.
Google.Maps.LatLng	Especifica a posição de um ponto definindo as coordenadas (Latitude e Longitude).
Google.Maps.Size	Especifica o tamanho dos marcadores no mapa.
Google.Maps.InfoWindow	Define a janela de informação de um marcador no mapa.
Geolocation.getCurrentPosition	Define a localização do utilizador no mapa.
Google.Maps.MarkerImage	Define a imagem que o marcador vai ter no mapa.
Google.Maps.DistanceMatrixService()	Estima a distância entre pontos no mapa.

Focando na área compreendida no centro da cidade de Lisboa, foram utilizadas coordenadas de ecopontos fictícios que serviram como teste para a implementação da geolocalização na aplicação. Todas estas localizações foram predefinidas e registadas na base de dados através das coordenadas estando disponíveis para os utilizadores as verem no mapa. Em algumas das localizações estão configuradas ilhas de ecopontos e noutras um só ecoponto. No caso das ilhas de ecopontos, o ponto do mapa é mostrado com um marcador multicor permitindo que o utilizador diferencie, facilmente, as ilhas dos ecopontos únicos.

Posto isto, foram detalhadas as funcionalidades permitidas aos utilizadores através da aplicação, que são as seguintes:

3.4.3.1 Filtragem do Tipo de Ecoponto

Foi disponibilizado ao utilizador a possibilidade de filtrar os ecopontos no mapa pela cor/material fazendo com que haja uma melhor perceção da localização do tipo de ecoponto que é pretendido.

3.4.3.2 Pesquisa e roteamento do caminho para o Ecoponto

Esta funcionalidade permite ao utilizador encontrar o caminho até ao tipo de ecoponto que pretende chegar com base na sua distância e no estado do ecoponto (se encontra cheio ou vazio) em tempo real através da interação do utilizador. Na página, após a pesquisa, é apresentado o caminho detalhado, via rodoviária, para que o utilizador chegue ao respetivo ecoponto. Foram encontrados os cinco melhores ecopontos do mapa (em termos de classificação média dada pelos utilizadores) e depois é calculada, através da função "*DistanceMatrixService*", a distância mais baixa entre os cinco ecopontos para depois ser calculado o caminho através da função "*route*" entre a localização do utilizador e o ecoponto com distância menor.

O utilizador consegue, assim, obter esta informação em tempo real com o objetivo de reduzir a quantidade de lixo que existe em certos ecopontos dividindo-o assim equilibradamente, não fazendo a acumulação que atualmente existe.

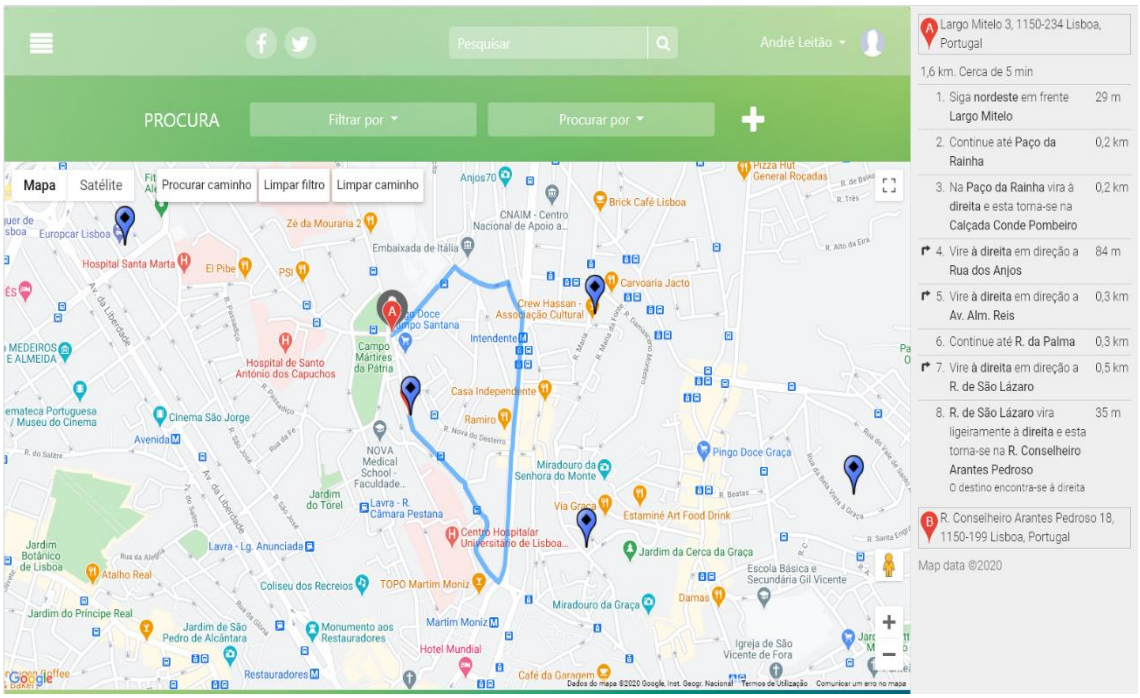


Figura 15 - Filtragem de ecopontos e rota para o ecoponto

3.4.3.3 Procurar caminho específico

Para além do caminho pela procura de um ecoponto específico, o utilizador pode saber a rota detalhada entre dois pontos escolhidos no mapa onde estão incluídas as coordenadas de todos os ecopontos e da localização do utilizador naquele exato momento. O cálculo da rota é feito pela função "*route*".

3.4.3.4 Adicionar Ecoponto

Quando o utilizador percebe que algum ecoponto não se encontra replicado no mapa da aplicação tem a possibilidade de adicionar ecopontos que existem nessas coordenadas, clicando num ponto específico do mapa faz com que seja adicionado o ecoponto permitindo que todos os utilizadores consigam ter acesso a ele.

Podemos ter vários ecopontos de cores diferentes (ilha de ecopontos) e consegue-se fazer essa adição no mesmo ponto do mapa, fazendo a seleção de todos os ecopontos existentes nessa localização. É possível, também, identificar se o ecoponto se encontra cheio ou vazio dando a sua classificação na altura da seleção.

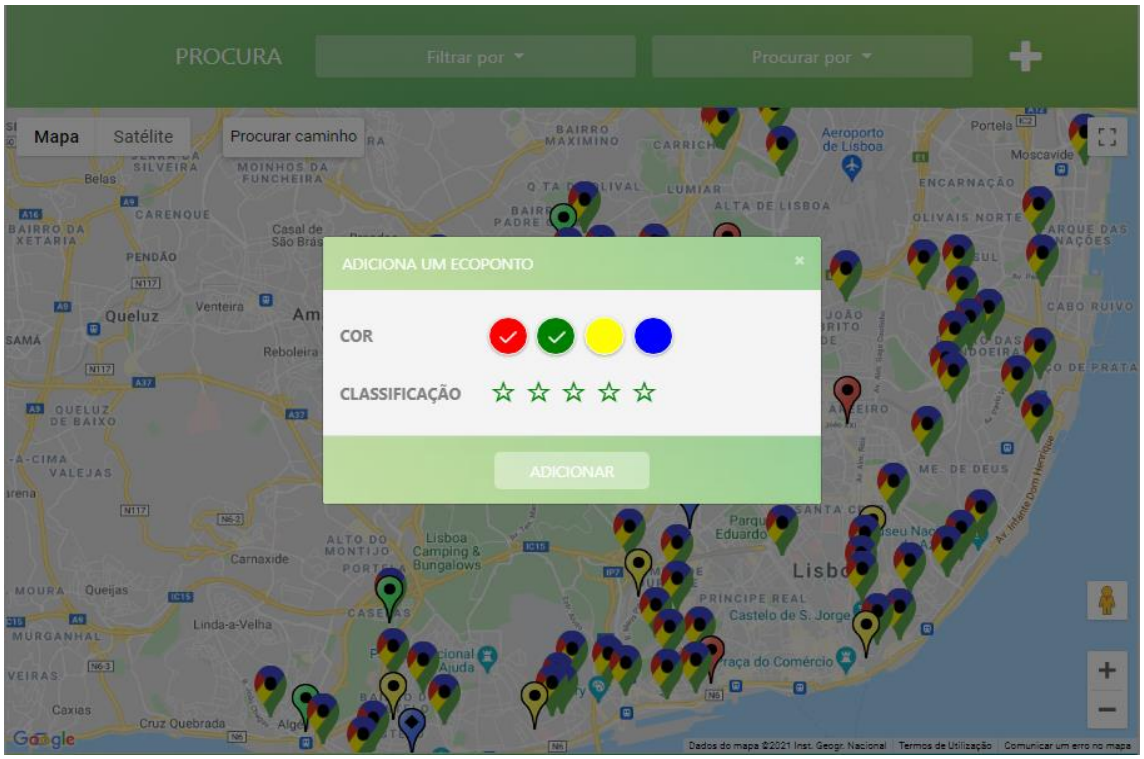


Figura 16 - Menu de adição de ecoponto

3.4.3.5 Mostrar estado do ecoponto

Para os utilizadores saberem o estado de cada ecoponto, ao passar sobre cada localização no mapa é possível verificar o menu que contém o estado do ecoponto através de uma pontuação de 1 a 5 estrelas, onde o 5 significa que o ecoponto está em muito bom estado, ou seja, pronto a usar e 1 significa que está cheio, pois não se encontra em condições para uso. A última atualização do estado do ecoponto aparece com o objetivo de informar o utilizador em tempo real, sendo que os estados que têm a data e hora mais recente são mais credíveis.

3.4.3.6 Classificação do ecoponto

Cada utilizador pode atualizar o estado dos ecopontos após o seu uso verificando assim o seu real estado. Isto ajuda a que todos os utilizadores tenham perceção do real estado dos ecopontos. No menu, onde é mostrada a classificação atual e última atualização, ao escolher o número de estrelas do ecoponto faz com que seja feita a sua atualização. Esta atualização é feita com base na média de todas as classificações que são feitas para aquele ecoponto específico.

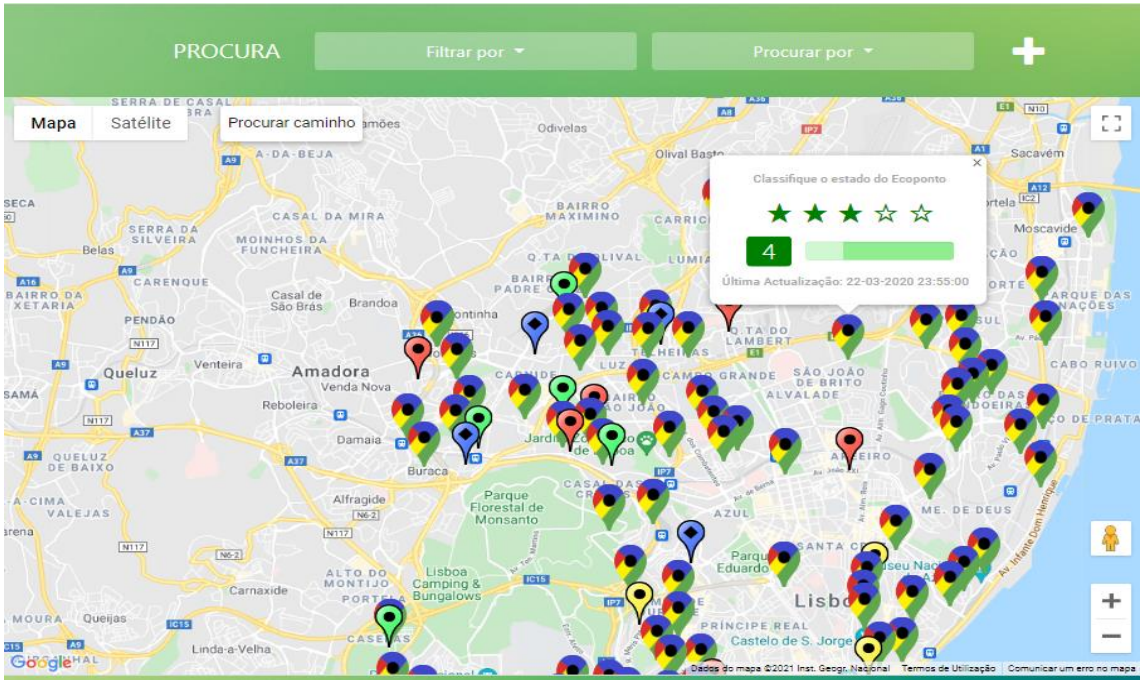


Figura 17 - Menu de classificação do estado do ecoponto

3.4.4 *Web Services*

O desenvolvimento do *Web Service RESTFul* em *Java* dividiu-se em dois serviços:

- Serviço *Google Maps*;
- Serviço *Facebook*.

Cada serviço desenvolvido contém vários métodos que irão ser processados após a comunicação com o cliente. Os métodos *HTTP* usados, fundamentalmente, nas chamadas dos serviços foram o *GET* e o *POST*, sendo o formato de dados usado o *JSON*.

Os métodos e as funções especificadas para cada um dos serviços implementados (*Facebook e Google Maps*) são representados na tabela seguinte.

Tabela 8 - Métodos dos serviços do Web Service

Serviço	Função	Método *HTTP*
Google Maps	Procurar Ecopontos	*GET*
Google Maps	Procurar Ecopontos por cor	*GET*
Google Maps	Procurar Top 5 de Ecopontos por classificação	*GET*
Google Maps	Procurar Ecoponto por referência	*GET*
Google Maps	Adicionar Ecoponto	*POST*
Google Maps	Atualizar Ecoponto	*POST*
Facebook	Adicionar Evento	*POST*
Facebook	Adicionar Utilizador *Facebook*	*POST*

A implementação destes serviços resultou da representação de diagramas de sequência para cada função onde podemos observar as ações desencadeadas pelo utilizador e o seu processamento entre aplicação, *Web Service* e base de dados.

Devido há quantidade de funções disponíveis no *Web Service*, foi especificado um dos diagramas de sequência e um excerto do código correspondente à função.

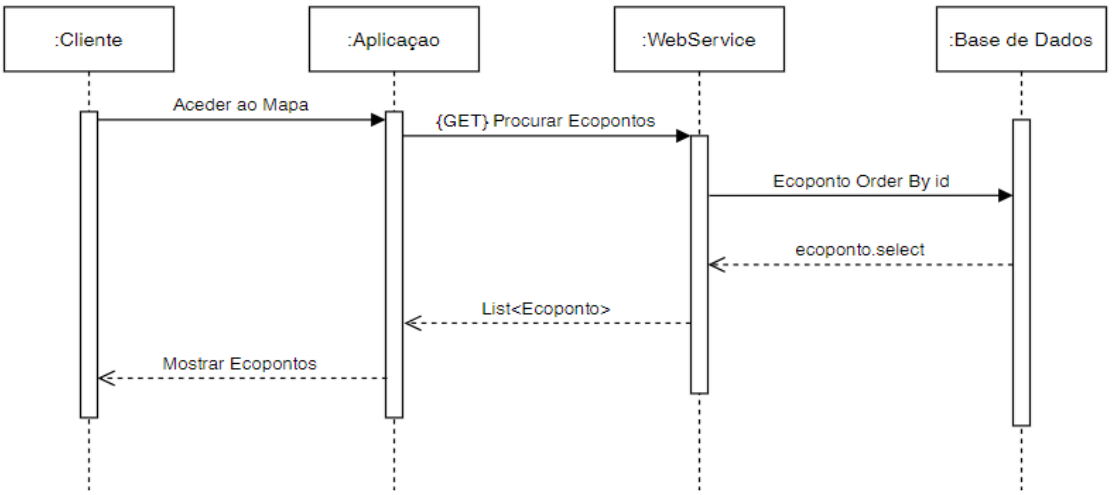


Figura 18 - Diagrama de sequência (método procurar ecoponto)

Código 2 - Chamada HTTP GET para procurar Ecoponto

```
var API_url = "HTTP://localhost:8080/EcopontoWS/REST/ecopontos/find";
$.AJAX({
      url: API_url, type: 'GET',
      dataType: 'JSON',
      crossDomain: true,
      success: function (data, textStatus, her) {}
});
```

A ligação entre as duas linguagens de programação, *Java* e *Python* é feita a partir de um *Web Service RESTFul* escrito em *Python* através da ferramenta *Flask*.

Este método foi usado para a classificação de imagens onde o utilizador faz *upload* de uma imagem, o servidor *web* obtém a imagem e envia-a para o *Web Service* como argumento (método *GET*) que ao receber processa-a e devolve o resultado da classificação do tipo de material ao servidor.

O serviço *Amazon S3* disponibiliza os ficheiros que são usados no *Web Service* para a classificação, em que num dos "*buckets*" irá ficar a imagem a ser classificada de forma temporária e o ficheiro *pikle* correspondente ao modelo final.

3.4.5 Interação com Serviços Externos

De modo a "enriquecer" esta aplicação com mais funcionalidades e tecnologias recorreu-se à integração de serviços externos com o servidor *web*. Um dos serviços utilizado e, por consequência o mais abrangente em funcionalidades disponibilizadas é apresentado em seguida.

- *Facebook for Developers:*

A plataforma *Facebook for Developers* oferece um conjunto de interfaces de programação e ferramentas que permite aos programadores integrarem diversas funcionalidades na sua aplicação. Para proporcionar uma maneira de as configurar ou solicitar acesso a API confidencias foi indispensável o registo da aplicação *web* na plataforma ajudando a distingui-la de outras aplicações.

Após o primeiro registo, é gerado um ID único para a aplicação sendo definida no modo de desenvolvimento, permitindo que sejam atribuídas funções que sirvam de auxílio para o desenvolvimento do projeto.

Quando estiver neste modo, a aplicação poderá:

- Receber permissões feitas pelos utilizadores que têm uma função na aplicação;
- Aceder aos dados dos utilizadores que têm uma função na aplicação ou aos dados de utilizadores de teste;
- Usar todas as permissões e produtos, bem como todos os recursos com exceção do acesso ao conteúdo público da página.

O modo "*live*" é obtido através da aprovação por parte da equipa de desenvolvimento do *Facebook*, mas que não foi possível obter e, por isso os testes foram feitos em modo de desenvolvimento. A implicação essencial ao ter a plataforma só no modo de desenvolvimento pretende-se pelo fato de só os utilizadores de teste registados na plataforma poderem usar as funções da API do *Facebook* e não todos os utilizadores que pretendam usar a aplicação.

Para que exista a comunicação entre a aplicação e a API do *Facebook* foi necessário definir os parâmetros seguintes que são gerados no momento do registo:

- *api-id*;
- *api-version.*

A API do *Facebook* faculta o *JavaScript SDK* que disponibiliza o código *JavaScript*, em que é necessário que estes parâmetros sejam definidos. As chamadas são feitas a partir de funções que são disponibilizadas na documentação da plataforma [29]. Temos como exemplo, a chamada de ligação entre a aplicação e a API de forma assíncrona usando uma função pré-definida.

Código 3 - Função assíncrona para comunicar com o Facebook for Developers

```
window.fbAsyncInit = function() {
      FB.init({
        appId      : '1191157747754766',
        cookie     : true,
        xfbml      : true,
        version    : 'v7.0'
      });
};
(function (d, s, id) {
    var JS, fJS = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
    JS = d.createElement(s); JS.id = id;
    JS.src = "//connect.Facebook.net/en_US/SDK.JS";
    fJS.parentNode.insertBefore(JS, fJS);
     }(document, 'script', 'Facebook-JSSDK')
 );
```

3.4.6 Aprendizagem Profunda (D*eep Learning*)

Para o desenvolvimento da classificação automática de imagens de resíduos, foi usada a linguagem *Python* pois tem bibliotecas disponíveis, tais como o *Keras, TensorFlow, FastAI* projetadas para permitir a experimentação rápida com redes neuronais profundas sendo fácies de usar e modelar.

O processo de classificação começa pelo desenvolvimento do modelo de treino dividido em várias iterações até obter o modelo final com os pesos das imagens treinadas e testadas, que será usado para a classificação.

3.4.6.1 *Dataset* de imagens

Foi usado um *dataset* pré-definido de imagens de materiais recicláveis retiradas da Internet [30] que são classificadas em cinco categorias (papel, cartão, vidro, metal, plástico), cada pasta da categoria respetiva contém em média 400 imagens o que faz com que seja um processamento mais lento, mas obtendo melhores resultados.

3.4.6.2 Tratamento de imagens

Antes do seu tratamento, foram organizadas as imagens em pastas de treino com as cinco categorias de cada material, em que as imagens do *dataset* são passadas para cada pasta da categoria correspondente. A partir da função "*ImageDataBunch*" da biblioteca *FastAI* é definido

que o modelo de treino vai usar 20% de imagens para teste e 80% para treino. O *batch size* é definido com o valor 16, este parâmetro representa o tamanho da amostra do número de imagens para processar em cada ciclo de treino. Para completar um ciclo, a função processará o número de etapas igual ao tamanho do *dataset* de treino divido pelo tamanho do *batch size* mais 1 para compensar qualquer parte fracionária [31].

Para compreender a diferença de performance do modelo de treino, em termos de resultados, são mostrados na tabela, em seguida, os valores de taxa de erro e tempo total de execução do modelo de treino, em vinte ciclos, para dois valores de *batch size*. O valor de *batch size* ideal seria 32, mas o processamento do modelo de treino fica muito lento, sendo o valor 16 o ideal obtendo, também, bons resultados em termos de taxa de erro.

Tabela 9 - Comparação resultados com batch size diferente

Batch Size	Taxa de Erro	Tempo Execução
16	5.65%	20 minutos
32	4.54%	1 hora e 33 minutos

Todas as imagens devem ter o mesmo tamanho sendo normalizadas alterando os valores de intensidade dos pixéis da imagem que ajuda a melhorar o resultado do modelo de treino.

Código 4 - Função ImageDataBunch

```
tfms = get_transforms(do_flip=True, flip_vert=True)
data = ImageDataBunch.from_folder(path,
                                  ds_tfms=tfms,
                                  valid_pct=0.2,
                                  bs=16)
data.normalize(imagenet_stats)
```

3.4.6.3 Modelo de treino (*ResNet*)

O modelo de treino é iniciado invocando a função "*CNN_learner*" do *FastAI*, em que é usada a arquitetura de rede neuronal baseada na Rede Neuronal Convolucional (*CNN*), sendo usada, neste caso a *ResNet34*. O modelo é visualizado através de um gráfico e é escolhido o melhor valor para ser usado nos ciclos de treino.

A função "*lr_find*" permite verificar, através de uma análise, o valor específico ou um intervalo de valores que minimiza a taxa de erro, o valor por defeito da função é de 3e-03, mas o valor escolhido foi de 5e-03 sendo este o valor ótimo para o processo de modelo de treino.

Este valor é escolhido quando a curva começa a estabilizar mesmo antes de começar a crescer. O valor de "*learning rate*" determina em que medida as informações recém-adquiridas substituem as informações antigas. Uma taxa de aprendizagem muito alta fará com que a

aprendizagem ande para valores mínimos, mas uma taxa com o valor muito baixo levará muito tempo a convergir ou ficará presa num mínimo local indesejável [32].

A Figura 19 mostra uma análise de como deve ser escolhido o "*learning rate*" e quais as opções em ter em conta na escolha. Os gráficos da figura estão disponíveis em [33].

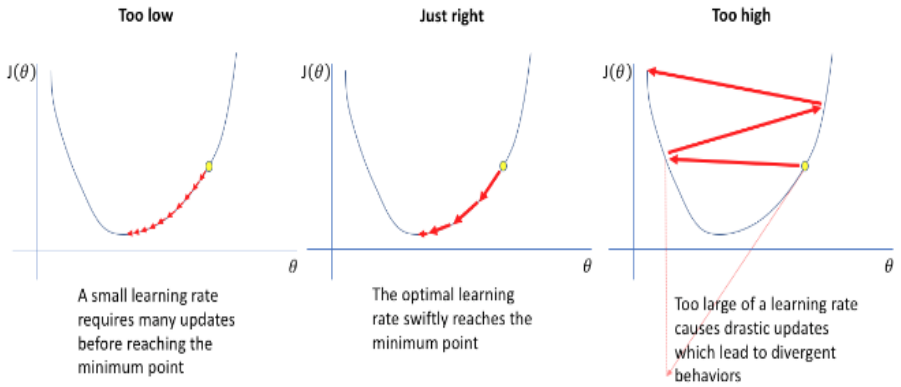


Figura 19 - Gráficos de learning rate

Código 5 - Função CNN_learner

```
learner = CNN_learner(data, models.resnet34, metrics=[accuracy])
learner.lr_find()
learner.recorder.plot()
```

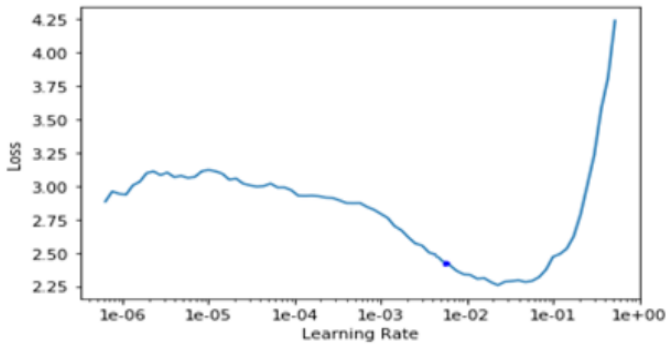


Figura 20 - Gráfico do modelo ResNet34

Apesar do valor ótimo escolhido, houve a necessidade de comparar a performance dos resultados do modelo de treino com um valor de "*learning rate*" intervalado, ou seja, foram escolhidos dois valores no gráfico mostrado anteriormente que resultassem num intervalo que fosse considerado ótimo.

Na tabela seguinte, são mostrados os valores de comparação entre o valor e o intervalo ótimo escolhido com diferentes *batch size*.

Tabela 10 - Comparação resultados com diferentes learning rate

Learning Rate	*Batch Size*	Taxa de Erro	Tempo Execução
5e-03	16	5.65%	25 minutos
6e-03 a 5e-03	16	6.7%	24 minutos
5e-03	32	4.54%	1 hora e 33 minutos
6e-03 a 5e-03	32	5.37%	1 hora e 31 minutos

3.4.6.4 Ciclos de treino

Uma vez que o modelo de treino se encontra definido, inicia-se o treino através da função "*fit_one_cycle*" do *FastAI* com o parâmetro 20, o que significa que serão executados vinte ciclos com o "*learning rate*" escolhido no passo anterior. A atualização da rede do modelo só ocorre no fim do processamento de cada bloco de imagens em cada ciclo efetuado. Treinar o modelo para vinte ciclos será suficiente para obter métricas de desempenho comparáveis com outros parâmetros. À medida que os ciclos são repetidos o acerto vai ser melhor e a taxa de erro cada vez menor. Após a execução dos ciclos, o modelo contém os pesos aprendidos e a arquitetura de rede para todas as camadas ligadas ao modelo.

No fim, é gravado o modelo final (construído utilizando todo o *dataset*) para depois ser usado no momento da classificação das imagens que são adicionadas pelos utilizadores na aplicação.

Código 6 - Função do Modelo de Treino

```
learner.fit_one_cycle(20,max_lr=5e-03)
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
interp.plot_top_losses(9, figsize=(15,11))
interp.most_confused(min_val=2)
doc(interp.plot_top_losses)
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
learner.export('trained_model_final.pkl')
```

epoch	train_loss	valid_loss	accuracy	time
0	1.271269	0.519753	0.786611	01:21
1	0.766586	0.392545	0.847280	01:19
2	0.651855	0.511090	0.815900	01:19
3	0.655425	0.523951	0.805439	01:20
4	0.651197	0.495665	0.849372	01:20
5	0.728554	0.482635	0.828452	01:20
6	0.689260	0.623334	0.786611	01:20
7	0.523001	0.540098	0.820084	01:20
8	0.535502	0.463289	0.853557	01:20
9	0.446356	0.366713	0.870293	01:19
10	0.419406	0.293478	0.903766	01:20
11	0.337107	0.321966	0.899582	01:20
12	0.339900	0.272028	0.912134	01:20
13	0.269808	0.183713	0.943515	01:20
14	0.236391	0.204871	0.939331	01:20
15	0.219416	0.191751	0.937239	01:20
16	0.193932	0.166818	0.949791	01:24
17	0.167360	0.181477	0.945607	01:20
18	0.128669	0.171754	0.949791	01:19
19	0.158596	0.170905	0.943515	01:20

epoch	train_loss	valid_loss	accuracy	time
0	1.171862	0.521436	0.803347	01:19
1	0.781515	0.528300	0.836820	01:17
2	0.698367	0.573451	0.820084	01:17
3	0.712025	0.539431	0.813808	01:17
4	0.732016	0.926391	0.738494	01:17
5	0.698246	0.743612	0.740586	01:17
6	0.655544	0.556409	0.826360	01:17
7	0.612008	0.469345	0.851464	01:17
8	0.546454	0.560770	0.836820	01:17
9	0.458838	0.314710	0.887029	01:17
10	0.451201	0.284724	0.882845	01:17
11	0.354866	0.257957	0.914226	01:17
12	0.306828	0.280270	0.895397	01:17
13	0.276289	0.234989	0.910042	01:17
14	0.235655	0.182138	0.920502	01:17
15	0.199726	0.164551	0.939331	01:17
16	0.206688	0.143562	0.939331	01:18
17	0.148482	0.148374	0.945607	01:17
18	0.134997	0.146708	0.943515	01:17
19	0.135713	0.163235	0.933054	01:17

Figura 21 - Resultados com valor ótimo (esquerda) e com intervalo ótimo (direita)

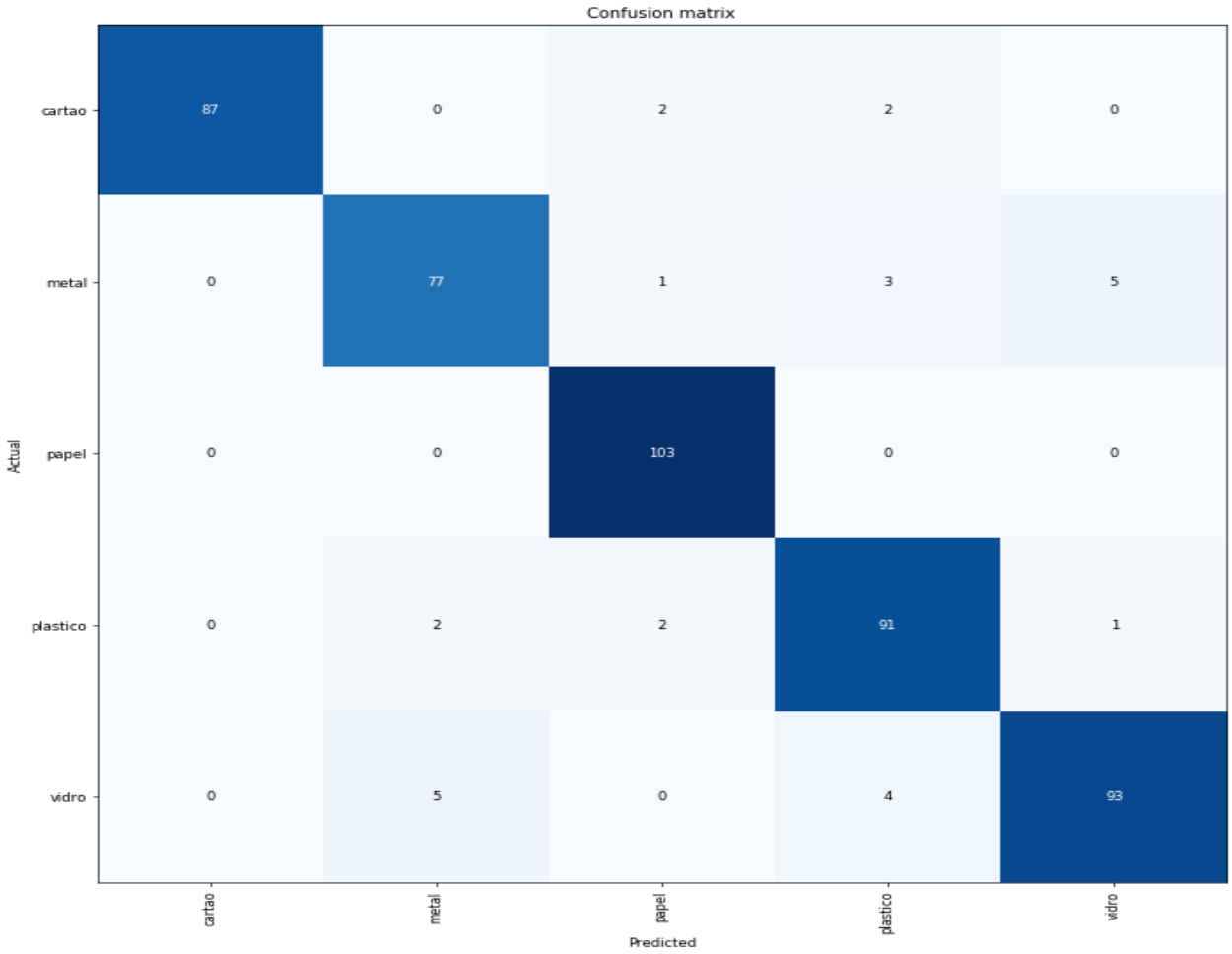


Figura 22 - Matriz de confusão do modelo treino

3.4.6.5 Resultados dos ciclos de treino

Podemos ver na Figura 21, a taxa de percas de treino e de teste, o valor de precisão e o tempo de duração de cada ciclo para o modelo de treino com valor ótimo (à esquerda da figura) e intervalo ótimo (à direita da figura). A Figura 22 apresenta a matriz de confusão dos 20% de imagens testadas.

O resultado da taxa de acerto foi de 94.3% e pela matriz de confusão verificamos que a maior parte foi corretamente classificada, sendo a taxa de erro de 5.7%.

3.4.6.6 Classificação das imagens

Para a classificação das imagens é usado o ficheiro (formato *pikle* [34]) exportado no processo do modelo de treino que contém os pesos e os atributos referentes às imagens de treino. O ficheiro *pikle* é lido e a imagem carregada pelo utilizador é classificada através de um valor de previsão para cada categoria do tipo de material, o valor maior será o da categoria classificada. Existe um limiar (*threshold*), que identifica se a classe é não identificável (se o valor de previsão for menor que o limiar) ou identificável numa das categorias (se o valor de previsão for maior que o limiar). Deste modo, são filtradas as imagens que não correspondem ao tipo de material das cinco classes definidas classificando-as como "Classe não Identificável".

Após a classificação da imagem por parte da aplicação ter sido efetuada, o utilizador tem a oportunidade de a validar ou mudar para a categoria correta do material no caso de a classificação ser errada

A validação das imagens, pelo utilizador, introduz vantagens no futuro modelo de treino que venha a ser construído, isto porque o *dataset* terá que ter uma correta categorização de cada uma das imagens inseridas em cada classe para o aperfeiçoamento dos resultados diminuindo a taxa de erro. Neste caso, a validação dos utilizadores é importante nesse melhoramento do modelo.

Como podemos ver na Figura 23, o utilizador pode confirmar a classe sugerida pela aplicação ou então corrigir para outra classe se achar que o resíduo não está na classe correta. Após a validação de uma imagem, esta será enviada para o sistema de ficheiros do *Amazon S3*, que contém o *dataset*, para o "*bucket*" da categoria correspondente.

Como exemplo, é apresentada a tabela seguinte com as possibilidades que podem acontecer no processo de validação e sua respetiva ação.

Tabela 11 - Processo validação da classificação

Classe previsão	Classe após validação	Ação
Vidro	Vidro	Imagem inserida na pasta vidro do dataset e disponível para próximo modelo de treino.
Vidro	Metal	Imagem inserida na pasta metal do dataset e disponível para próximo modelo de treino.
Vidro	Classe não identificável	Imagem que não será usada no modelo de treino e é removida.
Classe não identificável	Cartão	Imagem inserida na pasta cartão do dataset e disponível para próximo modelo de treino.
Classe não identificável	Classe não identificável	Imagem que não será usada no modelo de treino e é removida.

Estas são as hipóteses existentes no processo, em que as imagens que sejam classificadas como classe não identificável não serão consideradas para o próximo modelo de treino e os resíduos classificados como material reciclável irão ser considerados. O utilizador ganha pontos na aplicação à medida que vai inserindo cada imagem que é classificada como um material reciclável. Este processo está documentado e disponível na referência [35].

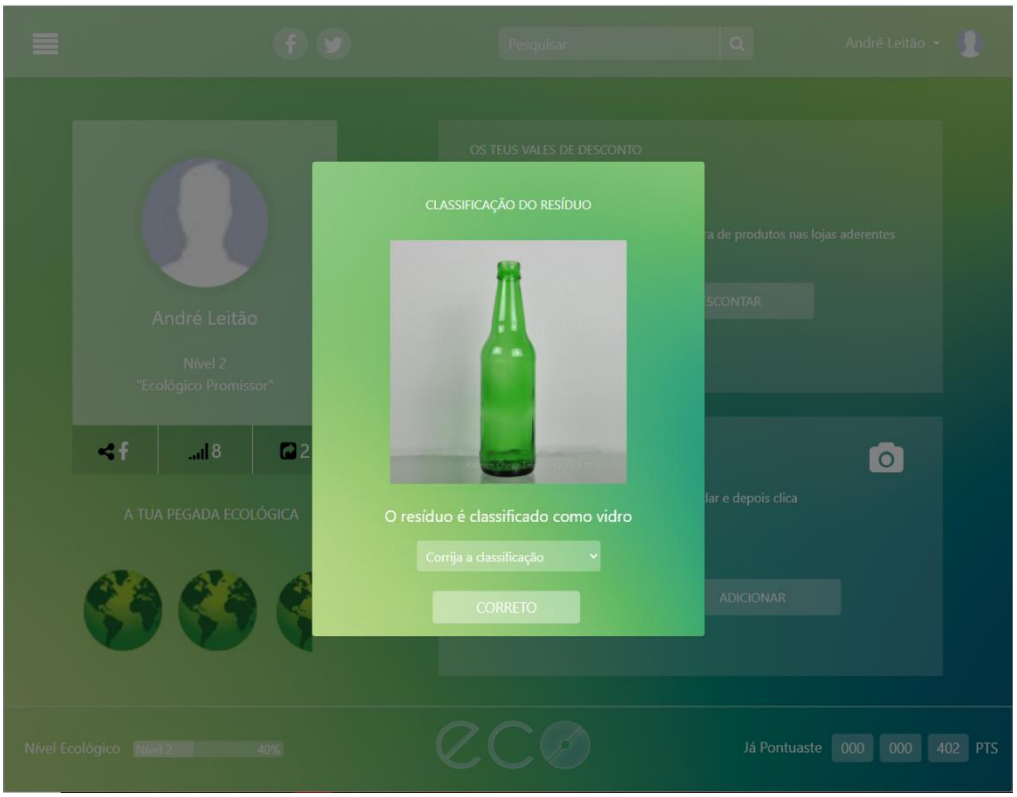


Figura 23 - Menu classificação do resíduo

3.4.6.7 Modelo de treino com imagens introduzidas pelo utilizador

Além do envio das imagens para a pasta correta, correspondente à categoria validada pelo utilizador, os dados das imagens são registados na base de dados com a referência da imagem, a classificação da previsão da classe, a classe atual validada pelo utilizador e o "*model_version*". Este valor irá identificar o número de imagens que ainda não foram usadas no processo de modelo de treino e que já foram validadas com uma classe categorizável.

Código 7 - Pseudocódigo da validação do resíduo

```
#define function to assign model version and update records
 model_version <- 0 #define default value for model version
 img_url <- image url
 temp_folder <- name of the folder on filesystem
 cat_folder <- name of the class folder on filesystem
 actual_class <- result of class classification
 non_class <- string with value "Classe não identificável"
 if actual_class != non_class:
   model_version <- -1 #define value -1
   copy(temp_folder, cat_folder, img_url) #copy image to right folder
   update(actual_class, model_version) #update database row
 else:
    remove(img_url) #remove image from temp_folder
```

No pseudocódigo anterior verificamos como é feito o processo de validação de uma imagem após a ação feita pelo utilizador. Todas as imagens identificadas com o valor -1 de "*model_version*" são aquelas que ainda não foram treinadas e que entrarão na próxima construção do modelo de treino.

Após obter o novo modelo de treino, o "*model_version*" será incrementado e atualizado na base de dados para as imagens que foram treinadas permitindo diferenciá-las das novas imagens introduzidas pelos utilizadores.

Na tabela seguinte são mostradas, como exemplo, duas classificações registadas na base de dados com imagens que ainda não foram usadas no modelo de treino.

Tabela 12 - Base de dados com duas imagens registadas

Id	*Filename*	*Predicted_class*	*Actual_class*	*Model_version*
1	vidro504.jpg	vidro	vidro	-1
2	metal598.jpg	plástico	metal	-1

Capítulo 4

Implementação do Modelo

Este capítulo descreve os métodos e tecnologias da implementação que foram utlizados ao longo do desenvolvimento da aplicação sendo apresentadas as configurações necessárias para o funcionamento da *EcoApp*.

A plataforma de trabalho usada para a realização deste trabalho foi o IDE Eclipse, onde é instalado e configurado o servidor *web Tomcat*. A partir deste momento, toda a implementação pode ser feita nesta ferramenta de trabalho.

Em primeira instância foi feita a realização do modelo de dados da aplicação.

4.1 Base de Dados

Foram definidas as tabelas e as suas associações consoante os detalhes que foram mencionados nos requisitos e casos de utilização no capítulo anterior. O modelo entidade-associação define um conjunto de entidades e associações numa base de dados e, para este projeto é descrito da seguinte forma.

A entidade “Utilizador” guarda os dados pessoais dos utilizadores registados na aplicação e é a entidade principal. O “Tipo_Utilizador” identifica se um utilizador é um utilizador comum ou uma entidade (restaurante, café, farmácia, etc.). O “Ecoponto” identifica os dados dos ecopontos e o “Tipo_Ecoponto” identifica a cor correspondente. As entidades “Link”, “Video” e “Imagem” guardam os *URL* de cada artigo, vídeo e imagem respetivamente. A tabela “Calendario” guarda todos os eventos do meio ambiente inseridos na aplicação, enquanto a tabela “Registo” faz o registo de todas as atividades realizadas pelo utilizador. Para guardar os níveis que são associados a cada utilizador é necessário criar a entidade “Nivel” e as imagens que são classificadas e validadas no processamento de reconhecimento de imagem são guardadas na tabela “Image_Class”. A entidade “Classificacao” guarda a classificação de cada ecoponto e a “Lista_Residuos” regista a listagem de resíduos que podem ser inseridos em cada ecoponto. Por fim, temos a tabela “Resposta”, “Pergunta” e “Questionario” que têm as perguntas dos questionários e guarda todas as respostas dadas pelos utilizadores.

Na Figura 24, é mostrado o modelo entidade-associação da aplicação.

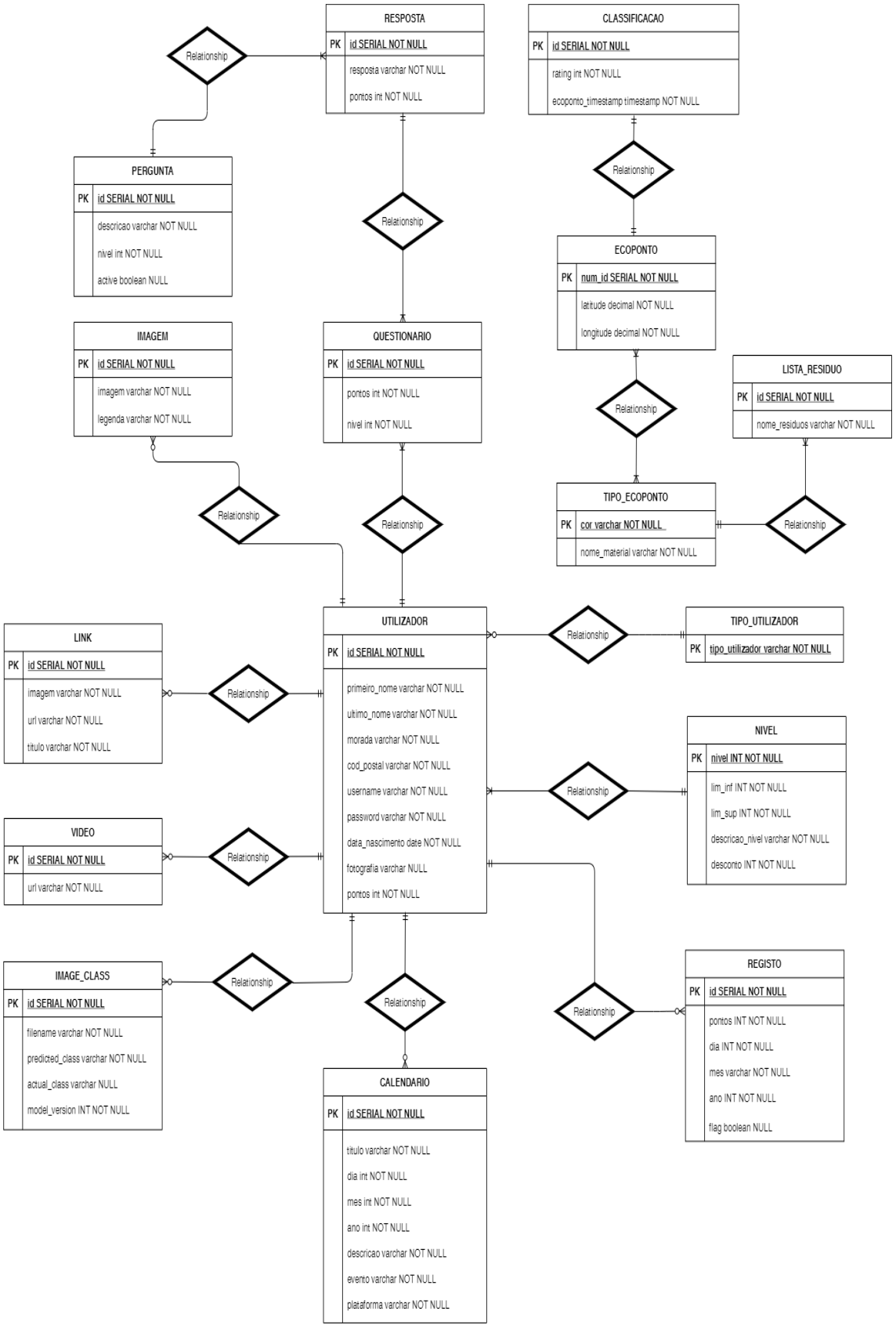


Figura 24 - Modelo Entidade-Associação

Para aceder à base de dados relacional, o *Hibernate* usa conectores ("*drivers*") *Java Database Connectivity (JDBC).*

Para integrar o *Hibernate* no servidor é necessário adicionar as bibliotecas referentes à ferramenta e o ficheiro *JDBC* para o *PostgreSQL*.

O *Hibernate* também requer um conjunto de configurações que são frequentemente colocadas num ficheiro de propriedades *Hibernate.properties* ou num ficheiro de configuração chamado *Hibernate.cfg.xml*. Algumas das configurações relativas à ligação do ficheiro de configuração com a base de dados são mostradas em seguida.

Código 8 - Ficheiro de configuração com os dados de ligação

```
<property name="Hibernate.connection.driver_class">org.PostgreSQL.
Driver</property>
<property name="Hibernate.connection.url">JDBC:PostgreSQL://ec2-54-
225-254-115.compute-1.amazonaws.com:5432/d2htjpt0nojbo4</property>
<property name="Hibernate.connection.CharSet">utf8</property>
<property name="Hibernate.connection.useUnicode">true</property>
```

O IDE Eclipse contém um *plugin* do *Hibernate* que pode ser integrado nas aplicações que estão a ser desenvolvidas. Esta integração permite o acesso à opção do *Hibernate Code Generation* que consiste em gerar as classes *Java* e ficheiros *XML* correspondentes às entidades da base de dados da aplicação. Este método facilita o desenvolvimento do código e permite ao programador ter todas as entidades e associações da base de dados representadas no servidor.

Após a criação da sessão e antes de fazer as transações, é imprescindível mapear todos os ficheiros *XML* no ficheiro de configuração do *Hibernate*. A partir deste ponto já é possível fazer as chamadas de *SQL* à base de dados desenvolvendo a query pretendida pelo programador.

A documentação da configuração do *Hibernate* está disponível na referência [36].

4.2 Camada de Controlo

Na camada de controlo o *Struts2* foi integrado com o IDE Eclipse e para o configurar foi necessário adicionar as bibliotecas correspondentes no servidor.

O *Struts2* usa o ficheiro *struts.XML* para especificar o relacionamento entre um *URL*, uma classe *Java* e a página de visualização *JSP*. Este ficheiro foi configurado para processar e redirecionar as ações definidas pelo programador dependendo do que o utilizador faça na aplicação desencadeando uma ação que irá redirecioná-lo para o processamento lógico da mesma. O processamento das ações é feito em classes *Java* através dos métodos usuais do *Struts2*.

Neste projeto, é usado o plugin do *JSON* disponível pelo *Struts2* com o objetivo de ter interação dinâmica após o resultado de uma ação feita pelo utilizador ou administrador.

Código 9 - Exemplo de uma ação no ficheiro struts.xml

```
<action name="addResiduo" class="com.actions.manageResiduo"
method="add">
    <interceptor-ref name="JSONValidationWorkflowStack">
    </interceptor-ref>
    <interceptor-ref name="defaultStack"></interceptor-ref>
    <result name="input" type="JSON">
        <param name="root">fieldErrors</param>
        <param name="wrapPrefix"><![CDATA[{ "errors" : ]]></param>
        <param name="wrapSuffix"><![CDATA[}]]></param>
    </result>
    <result name="success" type="JSON">/question.JSP</result>
</action>
```

O exemplo do código anterior indica que se o *URL* da página *JSP* tiver o nome de "*addResiduo*" vai redirecionar a aplicação para a página "*question.jsp*", em caso de sucesso, usando o plugin *JSON*. À medida que os *JSP* são criados, este ficheiro vai sendo preenchido com as ações que são necessárias ao projeto.

Para que a estrutura do *Struts2* funcione com a aplicação foi necessário adicionar e mapear uma classe de filtro, de referência à ferramenta, no ficheiro *web.xml* do servidor.

A configuração do *Struts2* e todos os seus passos estão descritos em [37].

4.3 Aplicação *Web*

Antes de iniciar o desenvolvimento da camada de visualização foi estruturada e organizada a arquitetura de conteúdos para o perfil de utilizador e administrador. Estes conteúdos vão ser expostos no *front-end*.

As figuras seguintes representam a estrutura de informação da aplicação para o utilizador e administrador.

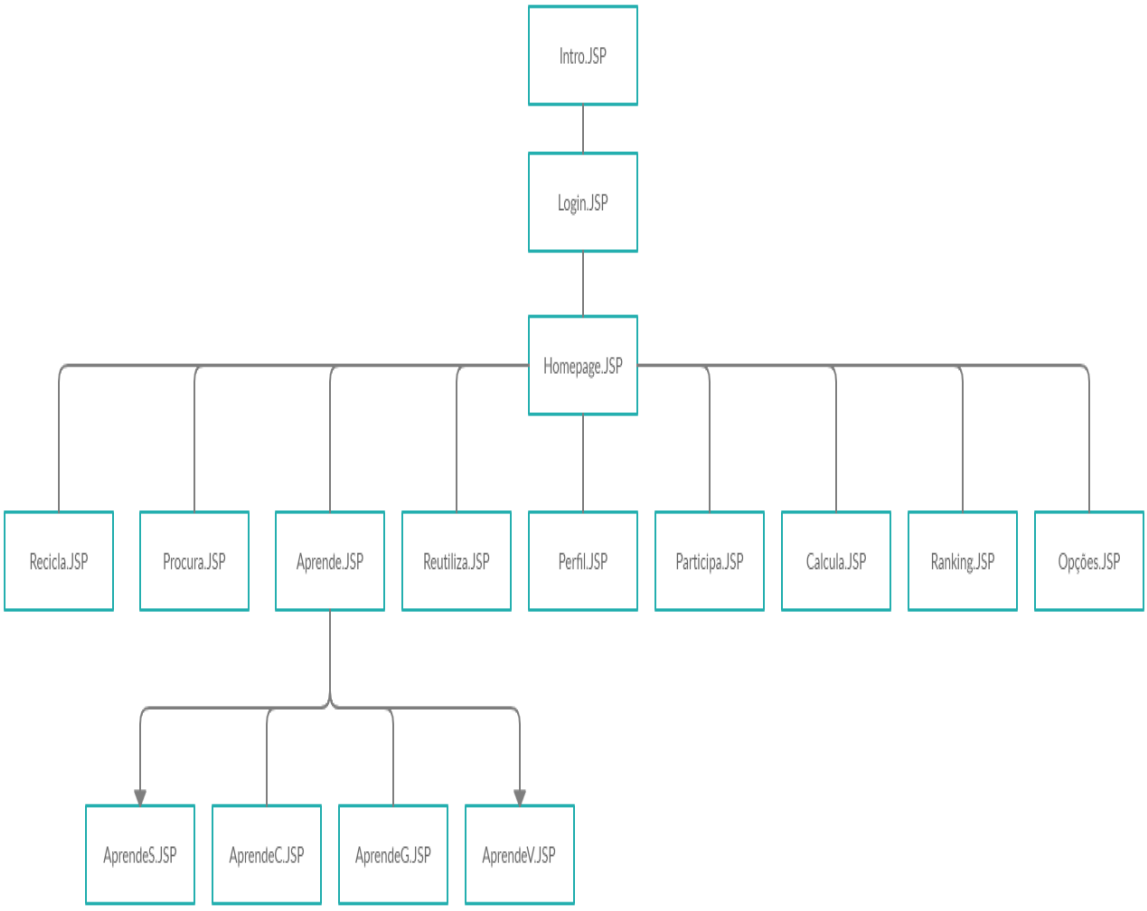


Figura 25 - Estrutura de conteúdos (Utilizador)

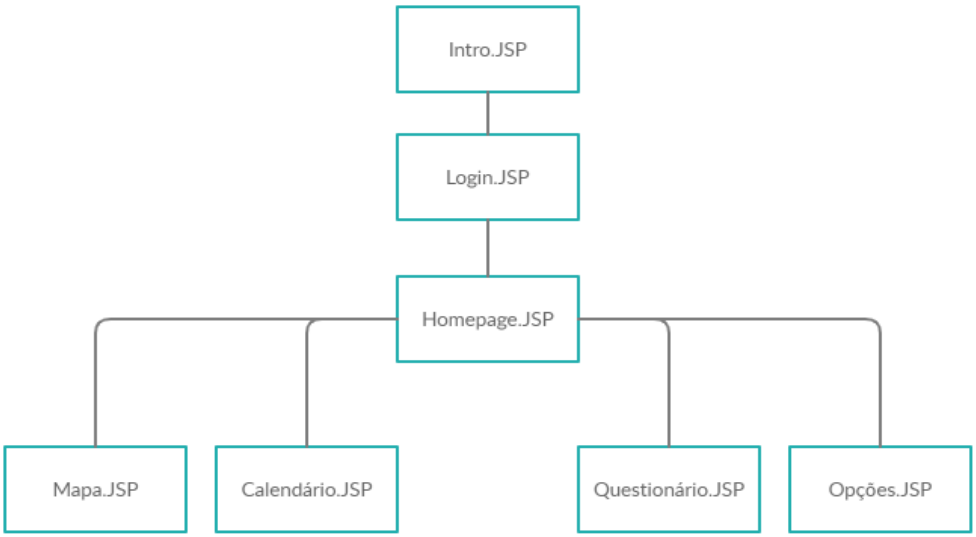


Figura 26 - Estrutura de conteúdos (Administrador)

Na página de *login* o utilizador, de forma dinâmica, pode registar-se, recuperar a *password* e fazer *login* (pela aplicação ou *Facebook*). Na página de *homepage* é possível aceder a todo o conteúdo definido na arquitetura de utilizador, inclusive às páginas da aplicação nas redes sociais *Twitter* e *Facebook*.

O administrador tem o mesmo método de *login* que o utilizador, mas os conteúdos são diferentes, visto que o administrador faz a gestão de conteúdos da aplicação.

4.3.1 Camada de Visualização (*Java* Server Page)

A camada de visualização contém as páginas *JSP* que são implementadas na aplicação para o utilizador visualizar o conteúdo.

Começamos pelo desenvolvimento das primeiras páginas *JSP* culminando com a abordagem do sistema de autenticação da aplicação, em que tanto para administrador e utilizador a arquitetura é igual.

4.3.1.1 Registo

O registo na aplicação é feito através de um formulário de preenchimento dos dados de utilizador sendo acedido dinamicamente através do *JavaScript*. Todos os dados introduzidos são validados pelo *HTML5*, mostrando uma mensagem de erro sempre que estiver incorretamente preenchido ou não preenchido.

O registo dos utilizadores tem a particularidade de ser feito como um utilizador comum ou como uma entidade, tais como, restaurantes, cafés, fábricas industriais, mercados, entre outros. A aplicação funciona de igual forma para cada tipo de utilizador.

4.3.1.2 Recuperar *Password*

O formulário de recuperação de *password* é acedido dinamicamente, em que o utilizador introduz o seu email e é gerada uma *password* aleatória pela aplicação, que é enviada num email a partir das configurações do servidor *SMTP* no *Mailgun*.

A configuração do servidor *SMTP* é feita com os seguintes dados:

- Host: *SMTP.mailgun.org*
- Porto: 587;
- Protocolo: *SMTP*.

Se o email não se encontrar na base de dados será mostrada uma mensagem de erro.

4.3.1.3 Sistema de Autenticação

Existem vários métodos para realizar o *login* que podem ser seguidos numa aplicação deste tipo, neste caso foi feita a opção de usar o *Struts2* juntando o conceito de criptografia inerente ao projeto. Para além do processo *login* foi implementado o de *logout.*

O *Struts2* usa um sistema de *login* baseado em sessões que usam intercetores. Os intercetores são responsáveis pela maior parte do processamento feito pela ferramenta de controlo, por exemplo, passar parâmetros para as classes de ação e disponibilizar sessões para as mesmas.

No ficheiro principal *struts.xml*, são definidas as ações que vão servir para a gestão de sessões de *login* dos utilizadores. Os intercetores definidos verificam se um utilizador está com a sessão ligada ou não. São definidos os "*globals results*" que quando a autenticação falhar o utilizador volta para a página de *login*. A ação de *login*, também é definida para os utilizadores e administrador da aplicação sendo redirecionados para as páginas correspondentes após o sucesso da autenticação. A configuração do sistema de *login* através do *Struts2* está disponível em [38].

É definido no ficheiro *web.xml* do projeto, o tempo limite de sessão e a página inicial da aplicação. Quando este tempo limite de inatividade é excedido, a aplicação volta à página de *login*. O administrador e o utilizador podem fazer *logout* e, após a sua saída, a sessão é removida.

No momento da autenticação, a *password* introduzida é cifrada e comparada com a *password* que se encontra na base de dados para o utilizador correspondente.

4.3.1.4 Redes Sociais

O *Facebook* e o *Twitter* foram as redes sociais utilizadas para interagir com a aplicação promovendo a sua utilização.

Depois da aplicação estar registada na plataforma de *Facebook* e em modo de desenvolvimento, poderão ser feitas as chamadas aos dados das permissões apresentadas em seguida:

- Acesso ao email do utilizador no *Facebook*;
- Nome do utilizador e da fotografia do utilizador no *Facebook*;
- Acesso aos eventos que se encontram, por exemplo, num grupo do *Facebook*;
- Acesso a grupos no *Facebook*.

Estas permissões permitem a integração das seguintes funcionalidades na aplicação.

- **Sistema de Autenticação via *Facebook*:**

A função de *login* foi disponibilizada através das funções do *JavaScript SDK* da API do *Facebook* onde obtemos os dados referentes aos utilizadores que estão a fazer o acesso (nome de utilizador, email, data de nascimento e fotografia). A autenticação é feita de igual forma, como se estivesse a entrar no *Facebook*, pelo email e *password*.

- **Eventos do Grupo do *Facebook*:**

A API do *Facebook* consegue obter todos os eventos do grupo referenciado pelo ID e pelo *token* associado à aplicação. Este grupo privado é criado e gerido pelo administrador da aplicação com o intuito de haver interação entre utilizadores via redes sociais. Neste grupo só se encontram os utilizadores registados na aplicação podendo ser adicionados pelo administrador no momento do registo deles.

Os eventos são mostrados na aplicação através do calendário ambiental e no grupo do *Facebook* referente à aplicação *EcoApp*.

- **Partilha no Grupo do Facebook:**

Através da aplicação é permitido a partilha de links *web* relacionados com a reutilização de materiais reciclados no grupo do *Facebook*, assim como os vídeos adicionados pelos utilizadores e administrador.

Na Figura 27 é mostrado um dos exemplos de interação com as redes sociais onde é possível a partilha dos vídeos no grupo do *Facebook*.

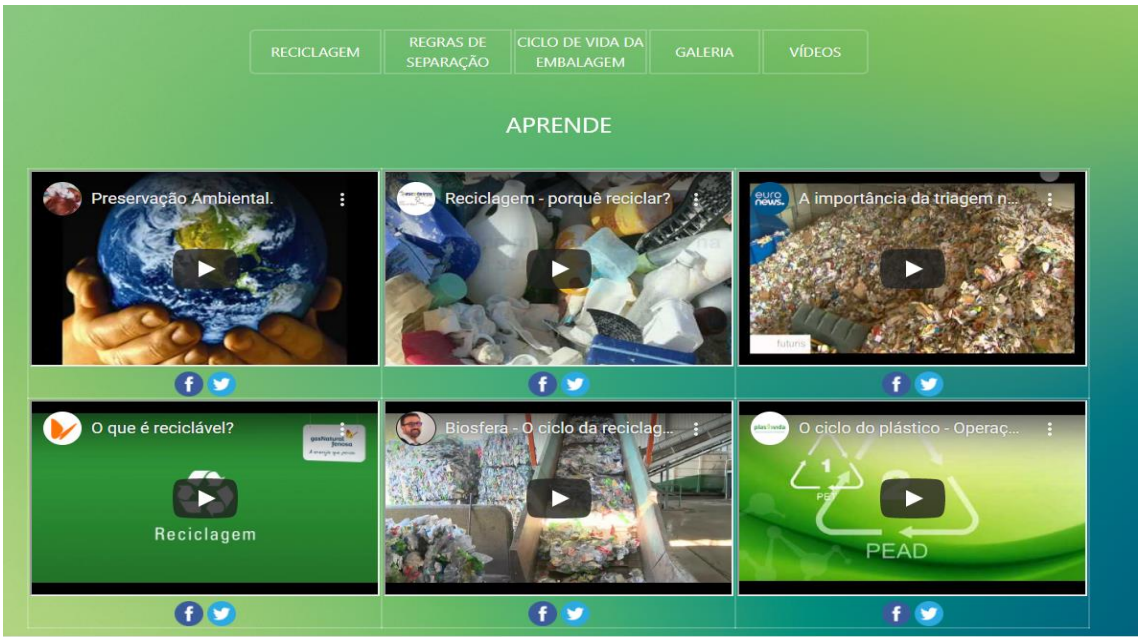


Figura 27 - Página de vídeos para a partilha nas redes sociais

Os dados do utilizador e os eventos do grupo são registados na base de dados através do *Web Service RESTFul*. Os eventos são atualizados à medida que são adicionados novos eventos no grupo privado do *Facebook*.

O *Twitter* é usado para a partilha de links e vídeos na página referente à *EcoApp* na rede social, não dispondo de qualquer funcionalidade integrada na aplicação.

4.3.1.5 Conteúdo Aplicação *Web* - Utilizador

O conteúdo da aplicação contém todas as funcionalidades que foram descritas nos requisitos e os aspetos de visualização que fazem parte da interação com o utilizador ou administrador.

Foi implementado um sistema de pontuação que permite ao utilizador ganhar pontos após determinadas ações. Cada intervalo de pontos é classificado por um nível, começando no nível 1 até ao nível 5, podendo o administrador fazer a gestão de níveis a seu gosto. A pontuação é aumentada mediante as seguintes ações:

- Resposta a um questionário em cada nível existente;
- Adicionar imagens de cada resíduo reciclado;
- Adicionar links sobre artigos de materiais reciclados reutilizados.

O utilizador poderá perder os seus pontos se a sua última atividade for feita há mais de 30 dias, sendo avisado previamente antes da retirada de pontos. O conceito desta aplicação tem como objetivo a atividade regular em termos de uso.

Esta atividade regular será recompensada quando for disponibilizado o desconto em compras numa das três empresas fictícias criadas para teste neste projeto. Quanto maior for o nível do utilizador maior é o desconto disponível, podendo ser usado uma única vez em cada nível. Identificamos agora cada um dos níveis e respetivo desconto na Tabela 13.

Tabela 13 - Tabela de níveis

Nível	Descrição Nível	Desconto
1	Ecológico Aprendiz	5%
2	Ecológico Promissor	10%
3	Ecológico Regular	15%
4	Ecológico Experiente	20%
5	Ecológico Perito	25%

Existem algumas páginas desenvolvidas na aplicação apenas de cariz informativo para o utilizador, apresentando assim alguma simplicidade a nível de desenvolvimento técnico e no processamento de dados.

- **Recicla:**

Esta página tem como objetivo informar os utilizadores da inserção de cada resíduo no tipo de material correspondente. Os tipos de material são identificados por cores onde é detalhada a lista de resíduos de cada um. Esta divisão do tipo de material pode gerar novos ecopontos no futuro.

- **Aprende:**

Esta página e suas subpáginas têm como objetivo fazer com que os cidadãos aprendem a fazer reciclagem e explicar como deverá ser feita. Apresenta vários conceitos sobre a reciclagem, a separação dos resíduos, o ciclo de vida de cada embalagem reciclada, quais os resíduos que devem ser inseridos em cada um dos ecopontos principais (Amarelo, Azul e Verde) e Ecocentros, apresenta uma galeria de imagens baseado no tema deste projeto e por fim, mostra uma galeria com vídeos. Os vídeos podem ser adicionados carregando ficheiros ou através de links do *Youtube*.

- **Reutiliza:**

Esta página consiste na publicação de links acerca do tema deste projeto, mais concretamente a reutilização de materiais reciclados, onde os utilizadores podem partilhar os seus links. Nesta página, é possível verificar todos os links do utilizador autenticado.

- **Ranking:**

Os rankings de pontos dos utilizadores a nível mensal e global são mostrados nesta página onde são divididos por entidades e utilizadores comuns. É mostrado o top 5 das entidades e utilizadores comuns tanto no ranking mensal e global sendo mostrados os pontos totais dos utilizadores e sua percentagem.

- **Participa:**

É desenvolvido um calendário ambiental que mostra todos os eventos do grupo do *Facebook* e os eventos adicionados na aplicação. Os utilizadores têm conhecimento das datas, da descrição dos eventos e o título do evento.

- **Calcula:**

É possibilitado ao utilizador responder a um questionário em cada nível onde há a eventualidade do utilizador ganhar pontos dependendo das respostas dadas. Cada resposta tem uma pontuação definida e após o questionário ser respondido o utilizador recebe uma pontuação final. Até passar para próximo nível não será possível responder ao mesmo questionário. Todas as perguntas e respostas encontram-se registadas na base de dados.

- **Opções:**

Nesta página, há a possibilidade de alterar os dados do utilizador autenticado e adicionar imagens e vídeos na aplicação.

- **Pesquisa:**

A pesquisa não é uma página específica, mas está disponível em todas as páginas, cujo objetivo é possibilitar ao utilizador fazer uma pesquisa do que pretende aceder na aplicação através da barra de pesquisa.

4.3.1.6 Conteúdo Aplicação *Web* - Administrador

O Administrador tem como objetivo fazer a gestão de conteúdos da aplicação em várias vertentes, nomeadamente:

- Gestão de Níveis;
- Gestão de Perguntas;
- Gestão de Tipos de Utilizador;
- Gestão de Tipos de Ecoponto;
- Gestão de Eventos;
- Gestão de Imagens e Vídeos;
- Gestão de Utilizadores;
- Gestão de Resíduos.

Para além da gestão de cada uma das entidades descritas há a possibilidade de alterar os dados do administrador, interagir com o *Google Maps* e ligar de forma interna com a base de dados da aplicação de forma local.

- **Questionário:**

Nesta página é permitido fazer a gestão de conteúdos de várias vertentes da aplicação. O administrador pode escolher cada nível disponível e alterar o conteúdo das perguntas nos questionários concedidos aos utilizadores. O resto das gestões mencionadas suportam o processo de adição, alteração e remoção de dados permitindo a alteração de conteúdos na interface do utilizador.

- **Calendário:**

A gestão de eventos é feita pelo administrador adicionando, atualizando ou removendo eventos que existam no calendário ambiental. Os eventos que vêm do *Facebook* são visíveis na aplicação, através de outra cor, e onde o administrador não os pode remover ou alterar pela aplicação, mas sim pelo grupo do *Facebook*. Ao selecionar uma data é necessário adicionar os parâmetros que validem o evento para ficar registado na aplicação.

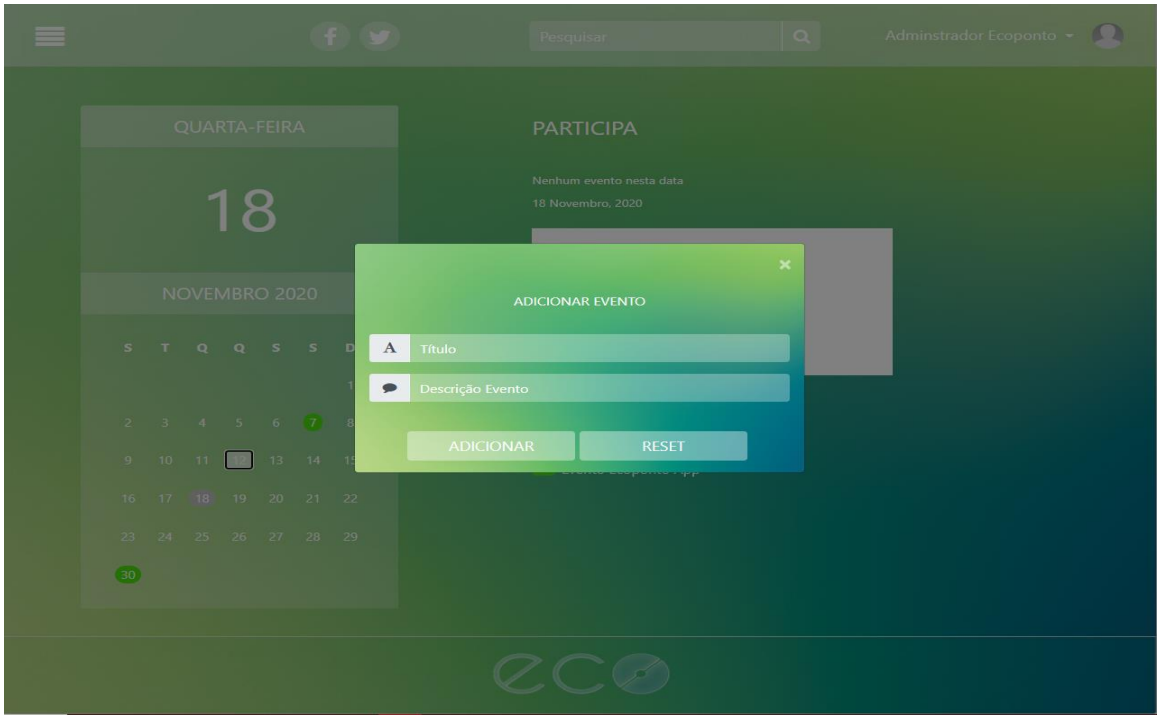


Figura 28 - Página Calendário no perfil de Administrador

O perfil de administrador contém menos conteúdo que o perfil de utilizador e, por isso, foram implementadas menos páginas *JSP*. Algumas das páginas são comuns com as do perfil de utilizador nomeadamente, a página de interação com o *Google Maps* e de alteração de dados e adição de imagens e vídeos.

4.3.2 *Web Services*

As aplicações com a arquitetura *REST* tem como uma das suas grandes vantagens a facilidade de desenvolver serviços para os clientes. Uma das principais *framework*s para desenvolver aplicações *REST* em *Java* é o *Jersey* que implementa todas as características da arquitetura *REST*.

Para configurar o *Jersey*, foi necessário adicionar os seus ficheiros de biblioteca no projeto e depois configurar o *web.xml* para que o *Servlet Jersey* intercete todas as chamadas para os serviços, por isso foi definido que as chamadas aos serviços fossem feitas com o prefixo /*REST*.

A implementação das classes expõe os métodos como serviços *REST*. A anotação *@Path* definida na classe, indica o endereço que o serviço será acedido. Para aceder a este método, é necessário indicar que se deseja aceder ao método com o tipo de resposta *APPLICATION_JSON*.

Os métodos *HTTP* são definidos tanto em *Java* como em *AJAX*, em que o caminho definido terá que ser igual para ambos possibilitando, assim, a sua comunicação. No código 10 e 11 é dado como exemplo a sintaxe de um dos métodos *GET* em *AJAX* e *Java*.

Código 10 - Sintaxe do método GET em Java

```
@GET
@Path("/find")
@Produces({ MediaType.APPLICATION_JSON })
public Response find() {
  try { (…) }
}
```

Código 11 - Sintaxe do método GET em AJAX

```
var API_url= "HTTPs://rocky-ridge05700.herokuapp.com/REST/ecopontos/
find";
$.support.cors = true;
  $.AJAX({
      url: API_url, type: 'GET',
      dataType: 'JSON',
      crossDomain: true,
      success: function(data, textStatus, xhr) {}
  });
```

Um dos aspetos importantes na configuração destes serviços é a serialização e deserialização de dados a partir do *Genson*. O principal objetivo desta biblioteca é fornecer métodos para serializar objetos *Java* para *JSON* e desserializar fluxos *JSON* para objetos *Java*. A maneira comum de usar esta biblioteca é ler *JSON* e mapeá-lo para algum *POJO* e vice-versa, ler o *POJO* e escrever *JSON*. Os dados que são usados nas várias chamadas aos serviços são mapeados para *POJO* para permitir a comunicação no formato de dados *JSON*. A configuração do *Genson* foi feita adicionando o ficheiro de biblioteca ao projeto.

Código 12 - Classe Ecoponto POJO com os dados em JSON

```
@JSONProperty("id")
private Integer id;
@JSONProperty("latitude")
private BigDecimal latitude;
@JSONProperty("longitude")
private BigDecimal longitude;
@JSONProperty("rating")
private int rating;
@JSONProperty("tipoEcopontos")
private List<TipoEcopontoPOJO> tipoEcopontos = null;
@JSONProperty("classificacoes")
private List<ClassificacaoPOJO> classificacoes = null;
```

4.3.3 Classificação automática de imagens de resíduos (*Deep Learning*)

A linguagem *Python* contém várias bibliotecas que permitem o desenvolvimento dos métodos relativos à classificação automática de imagens de resíduos.

No âmbito deste projeto, para fazer o processamento de classificação de imagem e de modelo de treino é usada a biblioteca *FastAI* [39] com o *PyTorch* [40].

O processo de classificação de imagens dividiu-se em duas fases:

- Modelo de treino;
- Classificação de imagens.

O processo de classificação de imagens iniciou-se pela configuração da ligação entre o servidor *Java* e o *Web Service RESTFul*, em que através do protocolo *HTTP* (método *GET*) foi permitido o envio e receção de dados. Este *Web Service* permite devolver ao servidor o resultado da classificação de uma imagem introduzida pelo utilizador. A biblioteca *FastAI* foi usada neste método com o intuito de obter a leitura do ficheiro do modelo de treino, anteriormente, desenvolvido.

Capítulo 5

Validações e Testes

Este capítulo tem como objetivo apresentar o resultado das validações e dos testes da aplicação e já disponível para exploração na *cloud* com um domínio definido para que os utilizadores consigam fazer os testes necessários para responder ao teste de usabilidade apresentado para este projeto, que vai avaliar a aplicação em relação à sua usabilidade em determinadas vertentes.

Um teste de usabilidade com utilizadores reais do produto pode rapidamente apontar as tarefas que as pessoas têm mais dificuldade no seu produto, mas ainda assim não consegue indicar o problema de usabilidade numa escala numérica. Para este projeto é usada a escala numérica de usabilidade, *SUS* [41], um método simples de averiguação do nível de usabilidade de um sistema. Os critérios que o *SUS* ajuda a avaliar são os seguintes:

- Efetividade (os utilizadores conseguem completar os objetivos?);
- Eficiência (o esforço e os recursos necessários para isso);
- Satisfação (a experiência foi satisfatória?).

O questionário consiste em 10 perguntas, e para cada uma delas o utilizador pode responder numa escala de 1 a 5, onde 1 significa Discordo Completamente e 5 significa Concordo Completamente. O ideal é que o teste do *SUS* seja aplicado ao final de um teste de usabilidade mais qualitativo, depois do utilizador tentar realizar um determinado grupo de tarefas usando a aplicação. Este método contém um cálculo de pontuação específico para determinar o resultado dos testes realizados pelos utilizadores.

Os testes de usabilidade foram divididos em três faixas etárias distintas onde, em cada uma delas, são realizados seis testes de usabilidade feitos por três homens e três mulheres, cujo objetivo é abranger uma opinião mais diversificada. As faixas etárias foram dividas entre menores de 30 anos, 30 a 45 anos e maiores de 45 anos.

- **Menos de 30 anos:**

Na tabela seguinte encontram-se os resultados obtidos por cada participante na faixa etária com idade menor a 30 anos e, por conseguinte, a análise gráfica dos resultados obtidos nos testes.

Tabela 14 - Resultados dos testes de usabilidade (Menor de 30 anos)

	Questões do *SUS*	**Respostas por Participante**					
		A	**B**	**C**	**D**	**E**	**F**
1	I think that I would like to use EcoApp frequently.	5	4	5	5	5	5
2	I found EcoApp unnecessarily complex.	2	1	2	1	1	2
3	I thought EcoApp was easy to use.	5	5	5	5	5	5
4	I think that I would need the support of a technical person to be able to use EcoApp.	1	1	1	1	1	2
5	I found the various functions in EcoApp were well integrated.	5	5	4	5	5	5
6	I thought there was too much inconsistency in EcoApp.	1	1	2	1	1	1
7	I would imagine that most people would learn to use EcoApp very quickly.	5	4	4	5	5	4
8	I found EcoApp very cumbersome (awkward) to use.	2	2	2	2	1	2
9	I felt very confident using EcoApp.	5	5	4	5	5	4
10	I needed to learn a lot of things before I could get going with EcoApp.	3	1	2	2	1	1

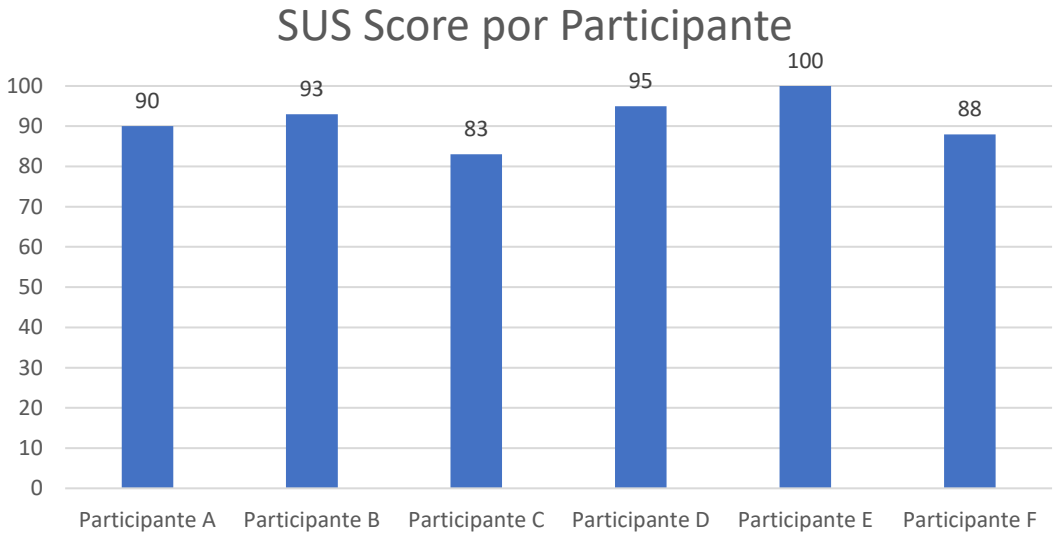


Figura 29 - Gráfico dos resultados dos testes de usabilidade (Menor de 30 anos)

O valor médio final da pontuação *SUS* é de 92, que representa a nota A (excelente) na tabela de classificação *SUS*.

- **Entre 30 e 45 anos:**

Os resultados obtidos dos testes de usabilidade e sua análise gráfica para a faixa etária entre 30 e 45 anos são mostrados em seguida.

Tabela 15 - Resultados dos testes de usabilidade (30 a 45 anos)

	Questões do *SUS*	**Respostas por Participante**					
		A	**B**	**C**	**D**	**E**	**F**
1	I think that I would like to use EcoApp frequently.	5	5	5	3	3	5
2	I found EcoApp unnecessarily complex.	1	1	2	1	1	2
3	I thought EcoApp was easy to use.	4	5	4	5	5	4
4	I think that I would need the support of a technical person to be able to use EcoApp.	1	2	1	1	1	1
5	I found the various functions in EcoApp were well integrated.	5	5	5	5	4	5
6	I thought there was too much inconsistency in EcoApp.	1	2	1	2	1	1
7	I would imagine that most people would learn to use EcoApp very quickly.	5	4	3	2	4	5
8	I found EcoApp very cumbersome (awkward) to use.	1	2	2	1	1	2
9	I felt very confident using EcoApp.	5	4	5	4	5	5
10	I needed to learn a lot of things before I could get going with EcoApp.	3	2	1	1	1	1

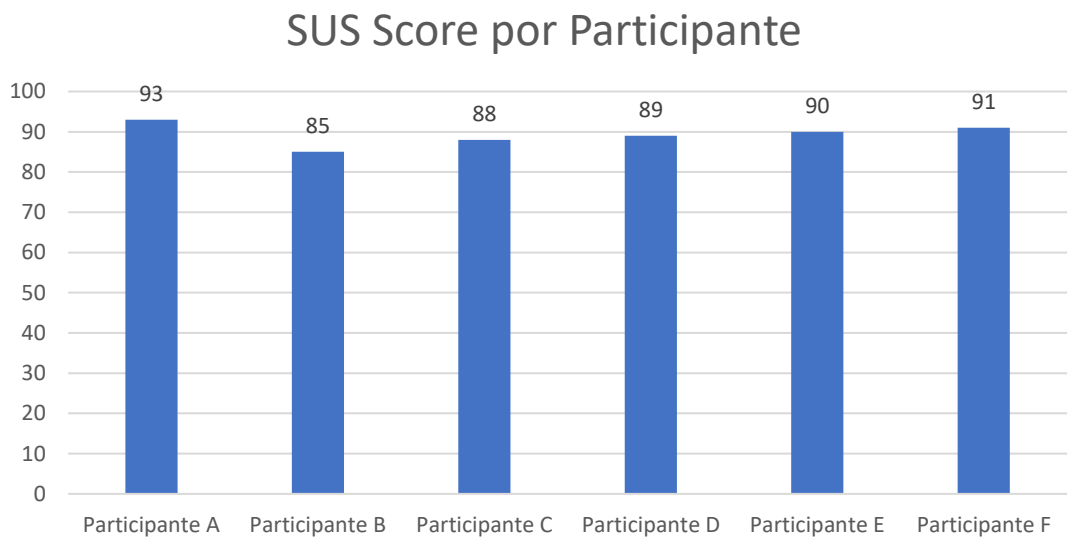


Figura 30 - Gráfico dos resultados dos testes de usabilidade (30 a 45 anos)

O valor médio final da pontuação *SUS* é de 89, que representa a nota A (excelente) na tabela de classificação *SUS*.

- **Mais de 45 anos:**

Os resultados obtidos dos testes de usabilidade e sua análise gráfica para a faixa etária de mais 45 anos são mostrados em seguida.

Tabela 16 - Resultados dos testes de usabilidade (Mais 45 anos)

	Questões do *SUS*	**Respostas por Participante**					
		A	**B**	**C**	**D**	**E**	**F**
1	I think that I would like to use EcoApp frequently.	4	4	5	5	4	4
2	I found EcoApp unnecessarily complex.	5	2	2	4	2	1
3	I thought EcoApp was easy to use.	4	3	3	3	3	5
4	I think that I would need the support of a technical person to be able to use EcoApp.	4	2	1	4	2	1
5	I found the various functions in EcoApp were well integrated.	5	5	5	4	4	5
6	I thought there was too much inconsistency in EcoApp.	3	1	1	2	1	1
7	I would imagine that most people would learn to use EcoApp very quickly.	5	3	5	4	3	4
8	I found EcoApp very cumbersome (awkward) to use.	2	2	2	3	2	2
9	I felt very confident using EcoApp.	4	4	4	4	4	5
10	I needed to learn a lot of things before I could get going with EcoApp.	3	2	2	4	1	1

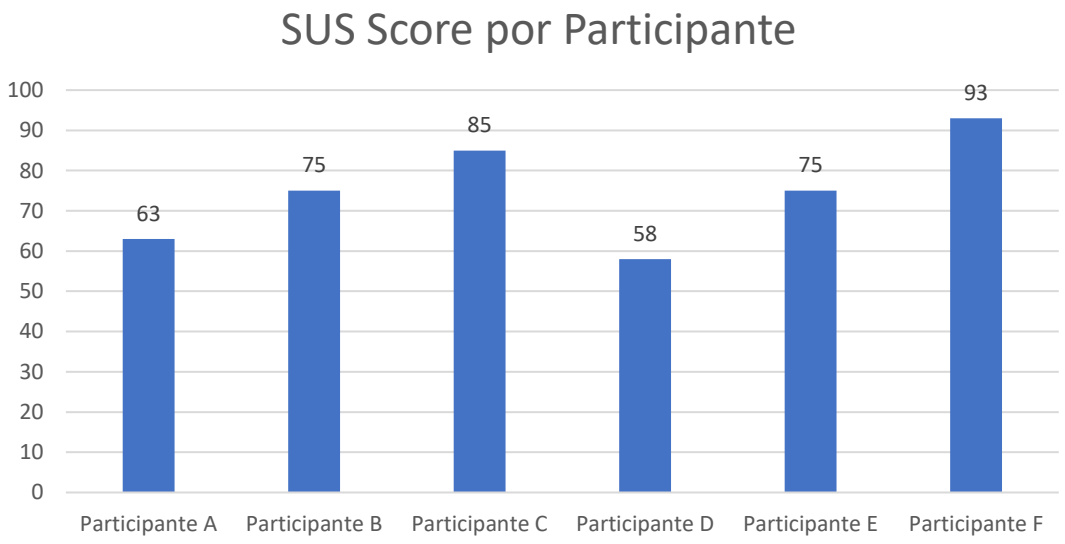


Figura 31 - Gráfico dos resultados dos testes de usabilidade (Mais de 45 anos)

O valor médio final da pontuação *SUS* é de 75, que representa a nota B (bom) na tabela de classificação *SUS*.

O resultado final global da pontuação *SUS* é de 85, sendo obtido, através da média de resultados finais das três faixas etárias, o que indica um excelente resultado nos testes realizados pelos utilizadores à aplicação *EcoApp*. Tendo em conta, os testes feitos por faixas etárias podemos verificar que as faixas etárias mais jovens não têm qualquer problema nenhum em usar a aplicação sem recurso a ajuda técnica, classificando-a como fácil e simples de usar sendo bem concebida para os utilizadores em termos de aprendizagem. A maior parte dos utilizadores que testaram a aplicação mostraram-se confiantes em usá-la no futuro.

A faixa etária mais velha tem algumas dificuldades na utilização da aplicação necessitando de ajuda para realizar as funcionalidades que estão presentes nela. Além disso, classificam a *EcoApp* como um bocado complexa necessitando de aprendizagem antes de a usar.

Em termos globais, os utilizadores assinalaram como comentário o fato de não ser possível fazer o *login* pelo *Facebook* devido à não autorização da plataforma para o mesmo, o que limita uma das funcionalidades da aplicação. Consideram ainda, que seria uma mais-valia a *EcoApp* estar disponível para *Android*. Como exemplo, foi realçado a parte de inserção de imagens para a classificação de resíduos onde pelo móvel é mais prático e direto de utilizar.

Na tabela seguinte são mostrados os resultados finais e respetivo *rating* classificado para cada uma das faixas etárias e, consequente, resultado global final.

Tabela 17 - Rating final dos testes de usabilidade

Faixa Etária	***Rating***
<30	A
>30 e <45	A
>45	B
Resultado Final	**A**

Capítulo 6

Conclusões e Trabalho Futuro

A aplicação *EcoApp* permite criar várias soluções que consciencializem e incentivem os cidadãos a contribuir de uma forma mais eficaz para a sustentabilidade do planeta.

Ao nível do enquadramento tecnológico, a utilização da arquitetura *MVC* facilitou o desenvolvimento desta aplicação visto que a separação entre a parte do cliente, responsável pela apresentação dos conteúdos ao utilizador e a parte do servidor, que processa e acede aos dados, permitiu o desenvolvimento independente de cada parte. Este desenvolvimento tornou possível que fossem realizados testes independentes, antes de passarmos para a fase seguinte, simplificando desta forma o ambiente de desenvolvimento e operação.

A possibilidade de alojar a aplicação num domínio de Internet permitiu que houvesse uma maior abrangência de acessos por parte dos utilizadores havendo a hipótese de consentir uma maior análise das validações e testes por parte dos mesmos, sendo possível assim a alteração de alguns aspetos considerados menos bons no projeto.

A realização deste projeto veio ao encontro dos objetivos propostos para o desenvolvimento *web* efetuado e tecnologias usadas, tendo sendo possível atingir vários conceitos que eram obrigatórios para a implementação desta aplicação. As funcionalidades presentes permitiram que fossem cumpridas todas as necessidades dos utilizadores com base no projeto anterior que deu origem a esta aplicação.

No decorrer do trabalho, não encontramos muitas limitações, apenas na parte da comunicação com uma das entidades ecológicas a nível da divulgação dos dados das localizações dos ecopontos da cidade de Lisboa. Podemos afirmar, que consideramos os objetivos cumpridos para esta aplicação implementada podendo o desenvolvimento deste projeto ser feito em outras plataformas expandindo o tema ecológico para contribuirmos para um planeta melhor e uma sociedade mais responsável pelas suas atitudes a nível ambiental.

Após o trabalho realizado identificam-se duas áreas para trabalho futuro, nomeadamente, na implementação desta aplicação numa plataforma móvel e na criação de um mecanismo que faça a atualização da base de dados da aplicação na perspetiva dos ecopontos. Para além destas duas áreas, há sempre a possibilidade de melhorar as funcionalidades gráficas da aplicação complementado a lógica desenvolvida.

No primeiro ponto mencionado, neste projeto foi desenvolvida uma aplicação *web*, mas pode, no futuro, ser implementado noutra plataforma, neste caso em *Android*. É importante que, para além da vertente *Web*, onde pode estar disponível para todos os dispositivos quando a aplicação é *responsive*, possa ser desenvolvido em *Android*, em que a integração é simplificada e pode ser usada de uma forma mais intuitiva e rápida.

No segundo ponto, tendo em conta que os ecopontos que foram utilizados no mapa do *Google Maps* foram de teste devido à impossibilidade de obter as coordenadas reais, deveria haver uma ligação com a Divisão Higiene Urbana em parceria com a Sociedade Ponto Verde para existir uma partilha dos dados dos ecopontos mantendo assim a aplicação sempre atualizada com dados reais tornando para o utilizador mais fácil a sua pesquisa e interação com o mapa.

Apêndice A

Arquitetura *MVC*

As aplicações informáticas implicam, tipicamente, a interação com o utilizador e o processamento interno de dados em função dessa interação. Para organizar a relação entre as classes responsáveis pela interação com o utilizador e as classes responsáveis para o processamento interno é feita uma separação em três camadas. A esta solução dá-se o nome de arquitetura *MVC* (*Model-View-Controller*). Esta arquitetura baseia-se na separação de três conceitos ligados entre eles, apresentação de dados e interação com os utilizadores (*client-side*) e por fim, os métodos que interagem com a base de dados (*server-side*).

A camada de modelo faz a ligação entre as outras duas camadas, consiste na parte lógica da aplicação que faz a gestão do processamento de dados através da lógica e das funções, ficando à espera da chamada das funções que permite o acesso para os dados serem gravados e exibidos.

A camada de visualização é onde os dados solicitados são visualizados. Esta camada tem interação direta com o utilizador provocando as suas ações que fazem depois a ligação com a camada que faz o controlo da aplicação.

As ações do utilizador são controladas na camada de controlo, esta camada faz a mediação entre a entrada e a saída de dados. Após uma chamada do utilizador, a camada de controlo faz a ligação com a camada de modelo para fazer a gestão dos dados consoante a ação pedida pelo utilizador no lado da camada de visualização.

Esta arquitetura é implementada funcionando como modelo principal de uma arquitetura geral dentro do processamento de dados da componente do servidor. O uso deste modelo beneficia a implementação de uma aplicação *web* com as tecnologias usadas em alguns aspetos nomeadamente:

- A arquitetura *MVC* suporta um desenvolvimento rápido e paralelo, quando desenvolvemos uma aplicação *web* é possível que um programador esteja a trabalhar na parte da visualização enquanto outro pode trabalhar na parte de controlo. Desta forma, a aplicação pode ser desenvolvida de forma mais rápida;

- A duplicação de código é muito limitada porque separa os dados, da parte da lógica e da visualização;
- Suporte mais fácil para novos tipos de cliente;
- A arquitetura *MVC* pode ser integrada no *JavaScript* oferecendo uma comunicação assíncrona que ajuda os programadores a desenvolver uma aplicação que carrega muito mais rápido;
- Para cada aplicação *web*, a interface do cliente tende a ser modificada mais frequentemente do que a parte lógica e de dados da aplicação. Em geral, é feita a modificação do aspeto das páginas *Web* como na cor, tipo de letra, *layout* do ecrã, etc. Adicionar uma nova página é muito fácil no padrão *MVC* porque a camada de modelo não depende da camada de visualização. Assim, nenhuma mudança no modelo irá afetar a arquitetura.
- O *MVC* retorna os dados sem formatação, portanto o mesmo conteúdo pode ser usado e chamado em qualquer interface. Qualquer tipo de dados pode ser formatado em *HTML*, mas também em *Macromedia Flash* ou *Dream Weaver*.
- As linguagens de *JavaScript* e *JQuery* podem ser integradas com o *MVC* para desenvolver aplicações *web* ricas em recursos.

O modelo *MVC* é certamente uma ótima abordagem para construir aplicações de *software* dada a sua estrutura fácil de implementar oferecendo inúmeras vantagens como as vistas acima. Projetos que são desenvolvidos com a ajuda deste modelo podem ser facilmente desenvolvidos com menos despesas e menos tempo. Acima de tudo, o poder de fazer a gestão de múltiplas visualizações torna o *MVC* o melhor padrão de arquitetura de desenvolvimento de aplicações *Web*.

Heroku e Domínio

Toda a implementação desta aplicação é feita num servidor que corre via localmente o que permite a sua testagem, desenvolvimento e o seu modo rápido de funcionamento. Para que os utilizadores tenham acesso a esta aplicação é necessário colocá-la na *web* através de um domínio.

Para proceder a este processo utilizamos a plataforma de *cloud Heroku* e o *Git* para o controlo de versões. O *Heroku* é uma plataforma na *cloud* que permite a integração de aplicações no domínio da *web*. Esta plataforma permite o suporte a diferentes linguagens de programação que simplifica o processo de integração, visto que não é necessário usar outro tipo de plataforma para completar o processo.

As integrações dos serviços do projeto dividiram-se em quatro vertentes principais, nomeadamente:

- **Implementação da Base de Dados:**

Em primeiro lugar, é feita a integração da base de dados na *cloud Heroku* onde é aproveitado o fato de esta *cloud* ter o suporte ao *PostgreSQL* o que facilita a incorporação desta no serviço.

Seguindo os passos descritos nas páginas de suporte da plataforma *Heroku* e disponíveis na bibliografia deste documento foi possível integrar a base de dados no *Heroku* com o *URL* pré-definido pela plataforma.

São mudados alguns parâmetros para a ligação entre a aplicação e a base de dados se manter, tais como, o *URL* da ligação com o novo nome da base de dados, o nome do utilizador e *password*. Todos estes parâmetros são facultados pela plataforma para configuração na aplicação.

- **Implementação do conteúdo do servidor *web*:**

Neste ponto, foi necessário efetuar algumas alterações no desenvolvimento do projeto para a integração no *Heroku*.

Começamos por transformar o projeto desenvolvido no *IDE Eclipse*, num projeto *Maven* [42] cumprindo assim um dos requisitos para a integração.

O *Maven* utiliza um ficheiro *XML* (*POM*) para descrever o projeto de *software* sendo referenciadas as suas dependências sobre módulos e componentes externos. O *Maven* descarrega bibliotecas *Java* e seus *plug-ins* dinamicamente de um ou mais repositórios armazenando-os numa cache local. Todas as dependências são definidas com base nas bibliotecas, que já existiam no projeto, antes de passar para o ficheiro *POM*.

Este projeto utiliza uma estrutura pré-definida ao qual terá de obedecer para ter o acesso a todos os recursos do projeto. Através dos procedimentos referenciados pelas páginas de configuração do *Heroku*, a integração do conteúdo do servidor *web* foi feita com sucesso na *cloud*.

- **Implementação do *Web Service RESTFul*:**

A implementação do *Web Service RESTFul* foi mais simples seguindo os mesmos passos que o anterior onde é transformado o projeto num projeto *Maven* e depois é feita a integração no *Heroku*. Um dos pontos importantes é a mudança dos vários *URL* que identificam os serviços do *Facebook* e do *Google Maps* nas respetivas chamadas de *GET* e *POST* onde serão substituídos pelo nome da aplicação fornecido pelo *Heroku* aquando da sua integração.

- **Implementação do *Web Service Python* em *Flask*:**

Para a integração do *Web Service* em *Python* é usado o *Gunicorn* que é um servidor compatível com o *Flask*. Este servidor permite a ligação do *Web Service* com o *Heroku*.

- **Criação Domínio:**

Após feitas todas as integrações no *Heroku* é adicionado um domínio à aplicação principal onde são configurados o *DNS Target* e o *CNAME* na plataforma que é usada para a criação dos domínios.

O domínio escolhido é o *WWW*.appecoponto.com adquirido na plataforma *Namecheap* e onde é configurado o *host* e o *DNS Target* que é obtido pelas configurações do domínio no *Heroku*. A partir das configurações definidas pelo *Heroku* foi fornecido um certificado *SSL* que permite a integridade para todas as solicitações da *web* para garantir que as informações sejam transmitidas com segurança.

Todas as integrações e configurações que foram feitas no *Heroku* são documentadas permitindo que seja seguido um processo que cumpre os requisitos do projeto e que está disponível em [43].

Apêndice B

Engenharia de Software - Requisitos

A gestão dos requisitos pode ser feita ou não através de prioridades. Independentemente do processo utilizado existem quatro atividades fundamentais num processo de Engenharia de requisitos:

- **Identificação de requisitos:** Identificação das fontes de informação do projeto e a descoberta dos requisitos deste;
- **Análise de requisitos:** Análise detalhada dos requisitos, perceção de conflitos e dependências entre estes. Nesta fase são identificados ainda os requisitos incompatíveis com o orçamento estipulado para o projeto. Como o próprio nome indica, é necessária a comunicação com o cliente, para a aprovar os requisitos propostos;
- **Especificação de requisitos:** Escrita dos requisitos de forma a ser percebida tanto pelos *stakeholders* do projeto como pela equipa de desenvolvimento;
- **Validação de requisitos:** Verificação da consistência e integridade dos requisitos. Esta atividade pretende certificar que os requisitos descrevem de forma aceitável o projeto a implementar, antes de estes servirem de base para o desenvolvimento do sistema informático.

As fases apresentadas anteriormente podem ser separadas e conjugadas de diferentes formas, dependendo do tipo de projetos e do processo implementado por cada organização que desenvolve a aplicação.

O Diagrama de Casos de Utilização especifica o comportamento e os aspetos envolventes do projeto em termos de casos de utilização e de atores. Este diagrama fornece um modo de descrever a visão externa da aplicação e suas interações com o mundo exterior, representando uma visão de alto nível da funcionalidade da aplicação.

Requisitos Funcionais

Os requisitos funcionais relacionados com a Gestão de Acessos são apresentados na tabela seguinte.

ID Requisito Funcional	Descrição Requisito Funcional
RF2.1	Acesso a utilizadores registados na aplicação.
RF2.2	Acesso a utilizadores que tenham conta no *Facebook.*
RF2.3	Utilizadores do tipo administrador devem ter um acesso diferente em relação aos utilizadores.
RF2.4	Disponibilizar a opção de recuperação de *password.*
RF2.5	Utilizador autenticado deve poder fazer *logout* em qualquer momento.
RF2.6	A aplicação deve disponibilizar ligação às contas de *Facebook* e *Twitter* da aplicação.

De forma a controlar o acesso dos utilizadores à aplicação, só os utilizadores devidamente registados e os utilizadores com registo no *Facebook* podem aceder à mesma.

A aplicação deve restringir o acesso ao painel de administração, a todos os utilizadores que não sejam do tipo administrador. No painel de administração, deve ser possível fazer a gestão de utilizadores, entre outras gestões que irão ser mostradas com a apresentação dos requisitos funcionais da aplicação no perfil de administrador.

ID Requisito Funcional	Descrição Requisito Funcional
RF3.1	Permitir a gestão de utilizadores
RF3.2	Permitir gestão dos níveis associados aos utilizadores
RF3.3	Permitir a adição de novos tipos de utilizador
RF3.4	Permitir a adição de novos tipos de ecoponto
RF3.5	Permitir a adição e remoção de perguntas em cada nível aos questionários da aplicação.
RF3.6	Permitir escolher as perguntas dos questionários de cada nível que devem ser mostradas na interface do utilizador
RF3.7	Permitir a gestão de imagens e vídeos
RF3.8	Permitir a gestão de eventos
RF3.9	Permitir acesso à base de dados da aplicação
RF3.10	Permitir o acesso à gestão dos ecopontos

Em seguida, apresentamos alguns dos diversos tipos de gestão da aplicação que existem, como a gestão de ecopontos, níveis, eventos, entre outros.

ID Requisito Funcional	Descrição Requisito Funcional
RF4.1	Permitir a gestão de Ecopontos tanto no perfil de Administrador como no Utilizador
RF4.2	Permitir a relação entre os ecopontos e o seu tipo de ecoponto
RF4.3	Permitir a gestão de níveis no perfil do Administrador
RF4.4	Suportar a gestão de eventos (adição, alteração e remoção) na aplicação no perfil de Administrador.
RF4.5	Permitir a gestão de eventos feita pelo Administrador no *Facebook*
RF4.6	Permitir a adição de links no perfil de Utilizador
RF4.7	Disponibilizar a partilha de links no *Facebook* e no *Twitter*
RF4.8	Permitir a adição de imagens e vídeos ao administrador e utilizador

O perfil de utilizador tem funcionalidades que permite aos utilizadores fazerem diversos tipos de ações que irão desencadear o processamento de dados no servidor. Na tabela seguinte são definidos os requisitos funcionais do perfil de utilizador.

ID Requisito Funcional	Descrição Requisito Funcional
RF5.1	Permitir a listagem de resíduos para cada tipo de ecoponto
RF5.2	Disponibilizar a informação de como separar os resíduos por ecoponto
RF5.3	Disponibilizar a localização, classificação e registos dos ecopontos
RF5.4	Suportar a pesquisa de ecopontos
RF5.5	Disponibilizar a informação dos eventos a realizar no perfil de utilizador
RF5.6	Permitir a resposta a um questionário em cada nível
RF5.7	Disponibilizar a informação dos links adicionados na aplicação
RF5.8	Permitir a resposta a um questionário em cada nível
RF5.9	Mostrar o ranking dos utilizadores registados na aplicação
RF5.10	Permitir o download de um documento PDF com um *QRCode* para descontos em lojas
RF5.11	Permitir o *upload* de imagens para o processo de classificação automático delas
RF5.12	Permitir a validação da classificação do resíduo na aplicação
RF5.13	A aplicação deve ter um sistema de pontuação para cada utilizador (entidade ou utilizador comum)
RF5.14	A aplicação deve ter um nível associado a cada utilizador baseado na pontuação

Por fim, são apresentados os requisitos funcionais que representam o processamento de reconhecimento de imagens introduzidas pelos utilizadores na aplicação.

ID Requisito Funcional	Descrição Requisito Funcional
RF6.1	Permitir a criação do modelo de treino através do *ResNet*
RF6.2	Processar o modelo de treino usando o *ResNet* com um dataset de imagens
RF6.3	Disponibilizar o ficheiro *pickle* para a classificação de imagens
RF6.4	Suportar o processamento de classificação automática de imagens de resíduos
RF6.5	Registar na base de dados as imagens classificadas com um tipo de material

Especificação Casos de Utilização

Neste anexo, são especificados alguns dos casos de utilização envolvidos no modo de funcionamento da aplicação *web*, nomeadamente, "Login", "Registo", "Aceder à Localização Ecoponto", "Selecionar Nível", "Consultar Eventos", "Alterar Dados Utilizador/Administrador" e "Aceder Redes Sociais".

Nome: *Login* **Descrição**: Permite ao Utilizador/Administrador fazer o *login* da aplicação **Atores**: Utilizador e Administrador
Cenário principal: 1. O utilizador/administrador acede à página de *login* da *EcoApp* 2. O utilizador/administrador introduz o nome e a *password*. 3. A aplicação verifica os dados. Dados válidos. Acesso à página de entrada da aplicação associada ao perfil do ator (utilizador ou administrador) e o caso de utilização termina. **Cenário alternativo 1:** 1. No passo 3 os dados são inválidos. 2. A aplicação mostra uma mensagem a informar o utilizador/administrador desse facto e o caso de utilização termina. **Cenário alternativo 2:** 1. No passo 2 fazer *login* pelo *Facebook* e o caso de utilização termina.

De seguida, são indicadas as operações associadas ao caso de utilização “Registo”.

Nome: Registo **Descrição**: Permite ao Utilizador fazer o registo na aplicação **Atores**: Utilizador
Cenário principal: 1. O utilizador acede à página de *login* da aplicação e clica no botão “REGISTA-TE” 2. O utilizador introduz todos os dados pedidos. 3. A aplicação verifica os dados inseridos. Dados válidos. 4. Após os dados serem válidos clica no botão “Registar” e o caso de utilização termina. **Cenário alternativo 1:** 1. No passo 3 os dados são inválidos. 2. A aplicação mostra uma mensagem a informar o utilizador desse facto e o caso de utilização termina.

No exemplo apresentado a seguir, são indicadas as operações associadas ao caso de utilização “Aceder à Localização do Ecoponto”.

Nome: Aceder à Localização do Ecoponto **Descrição**: Permite ao Utilizador/Administrador aceder à localização dos ecopontos através do *Google Maps* **Atores**: Utilizador e Administrador
Cenário principal: 1. O utilizador/administrador encontra-se na página de *homepage* da aplicação *web* após o *Login* 2. O utilizador/administrador abre o separador para escolher a opção desejada e seleciona a opção “Procura”. 3. A aplicação faz a ligação com o *Google Maps* e mostra todas as localizações dos ecopontos adicionados na Base de Dados e o caso de utilização termina.

As operações associadas ao caso de utilização "Selecionar Nível" encontram-se especificadas no exemplo seguinte.

Nome: Selecionar Nível **Descrição**: Permite ao Administrador selecionar o nível para fazer a gestão da aplicação *web* em várias vertentes **Atores**: Administrador
Cenário principal: 1. O administrador encontra-se na página de *homepage* da aplicação *web* após o *login* 2. O administrador abre o separador para escolher a opção desejada e seleciona a opção "Questionário". 3. O administrador seleciona o nível que pretende e pode fazer a gestão de perguntas do nível que escolheu e outro tipo de gestões e o caso de utilização termina.

As operações associadas ao caso de utilização "Consultar Eventos" encontram-se especificadas no exemplo seguinte.

Nome: Consultar Eventos **Descrição**: Permite ao Utilizador/Administrador consultar os eventos ambientais **Atores**: Utilizador e Administrador
Cenário principal: 1. O utilizador/administrador acede à página de *homepage* da aplicação após o *login* 2. O utilizador/administrador abre o separador para escolher a opção desejada e seleciona a opção "Calendário". 3. O utilizador/administrador acede à página dos eventos e clica num dos eventos que está assinalado no calendário. 4. A aplicação mostra o evento escolhido e o caso de utilização termina.

No exemplo apresentado a seguir, são indicadas as operações associadas ao caso de utilização "Aceder Redes Sociais".

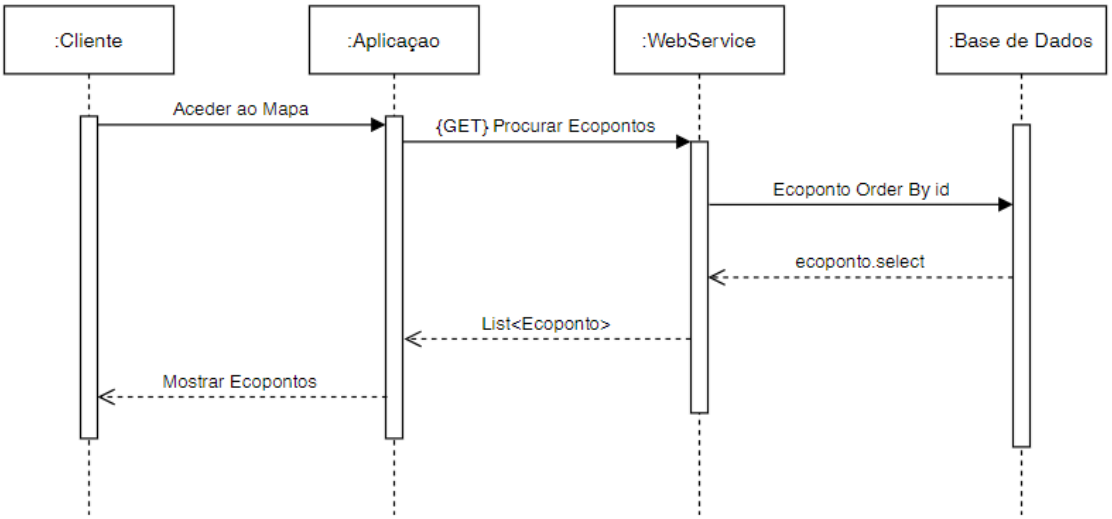
Nome: Aceder Redes Sociais **Descrição**: Permite ao Utilizador/Administrador aceder às redes sociais *Facebook* e *Twitter* **Atores**: Utilizador e Administrador
Cenário principal: 1. O utilizador/administrador encontra-se na página de *homepage* da aplicação *web* após o *login* 2. O utilizador/administrador carrega no botão com o símbolo do *Facebook* ou *Twitter* 3. A aplicação faz a ligação para a página da aplicação *web* na rede social e o caso de utilização termina. **Cenário alternativo 1:** 1. O acesso às redes sociais encontra-se em todas as páginas da aplicação *web.*

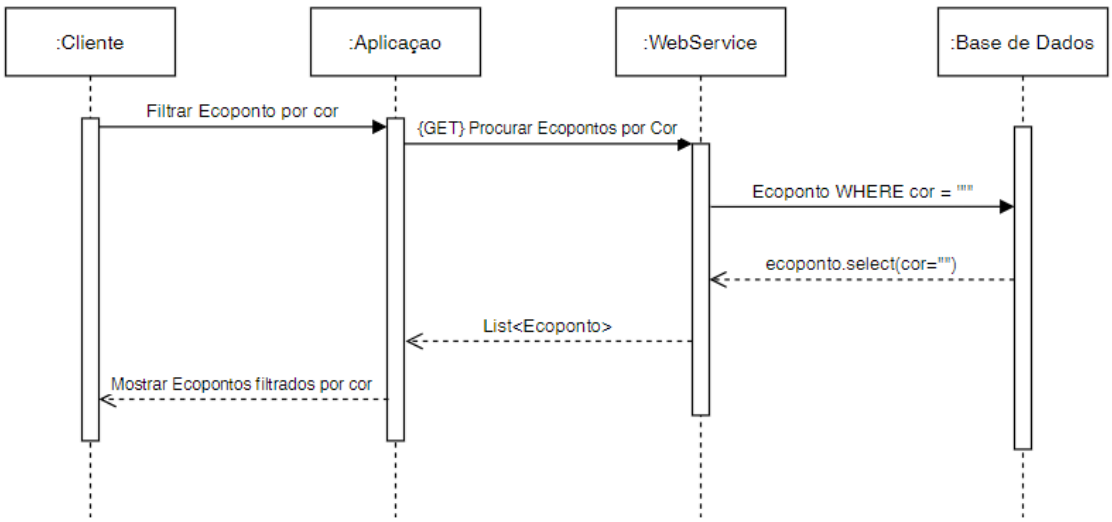
A tabela, em seguida, mostra as operações associadas ao caso de utilização "Alterar Dados Utilizador/Administrador".

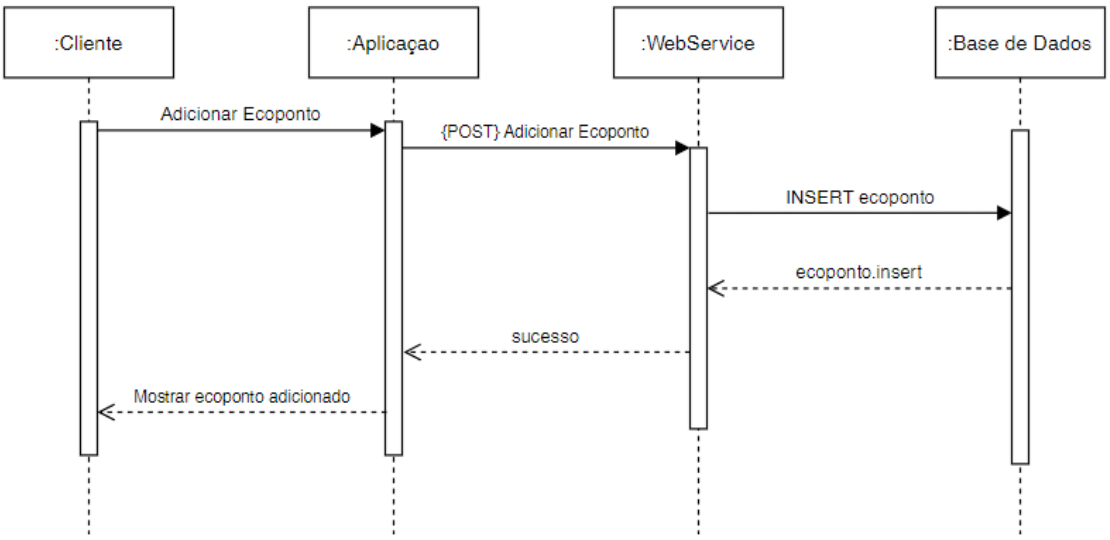
Nome: Alterar Dados Utilizador/Administrador **Descrição**: Permite ao Utilizador/Administrador alterar os seus dados pessoais **Atores**: Utilizador e Administrador
Cenário principal: 1. O utilizador/administrador encontra-se na página de *homepage* da aplicação *web* após o *login* 2. O utilizador/administrador abre o separador para escolher a opção desejada e seleciona a opção "Opções". 3. Na página de alteração de dados, o utilizador/administrador insere os dados. 4. A aplicação verifica os dados inseridos. Dados válidos. 5. Após os dados serem válidos clica no botão "ATUALIZAR" e o caso de utilização termina. **Cenário alternativo 1:** 1. No passo 4 os dados são inválidos. 2. A aplicação mostra uma mensagem a informar o utilizador desse facto e o caso de utilização termina.

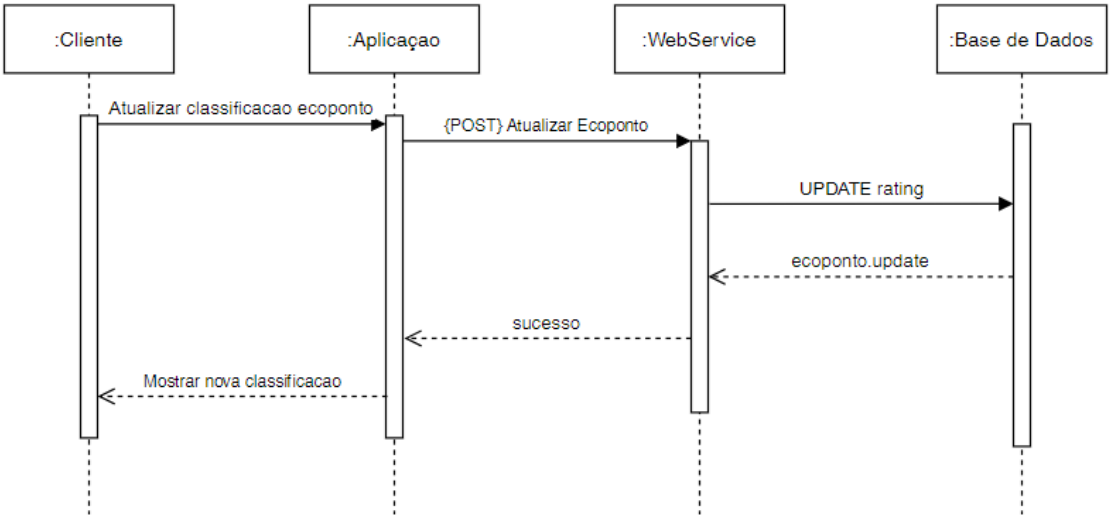
Diagramas de Sequência

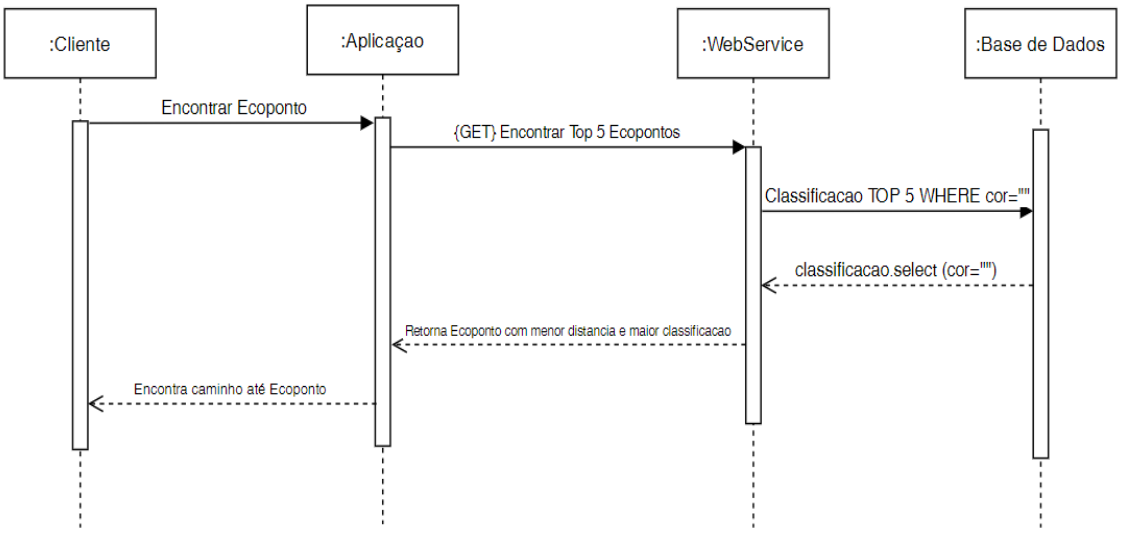
São representados os diagramas de sequência para cada função dos serviços relativos ao *Web Service* implementado em *Java*.

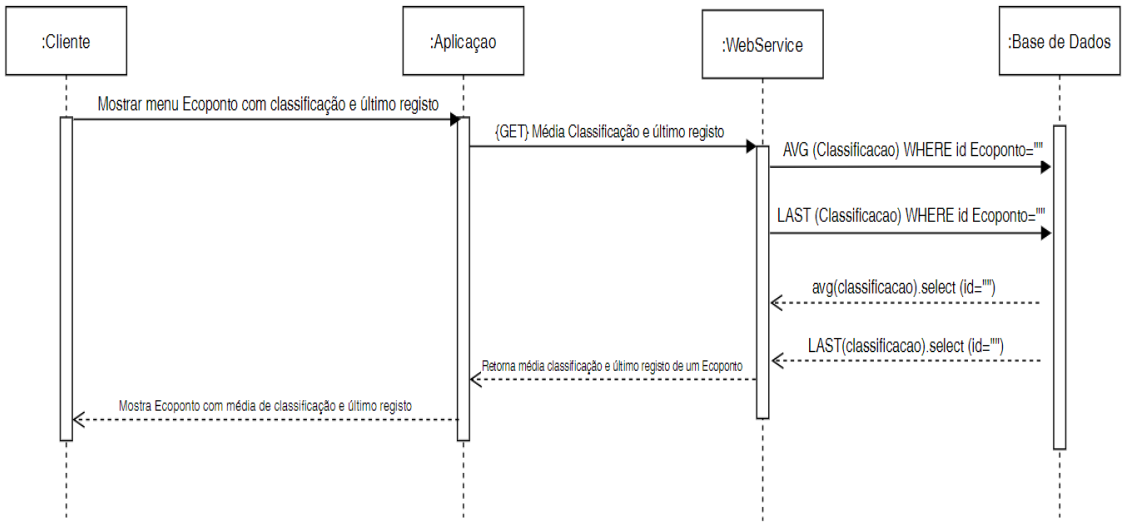


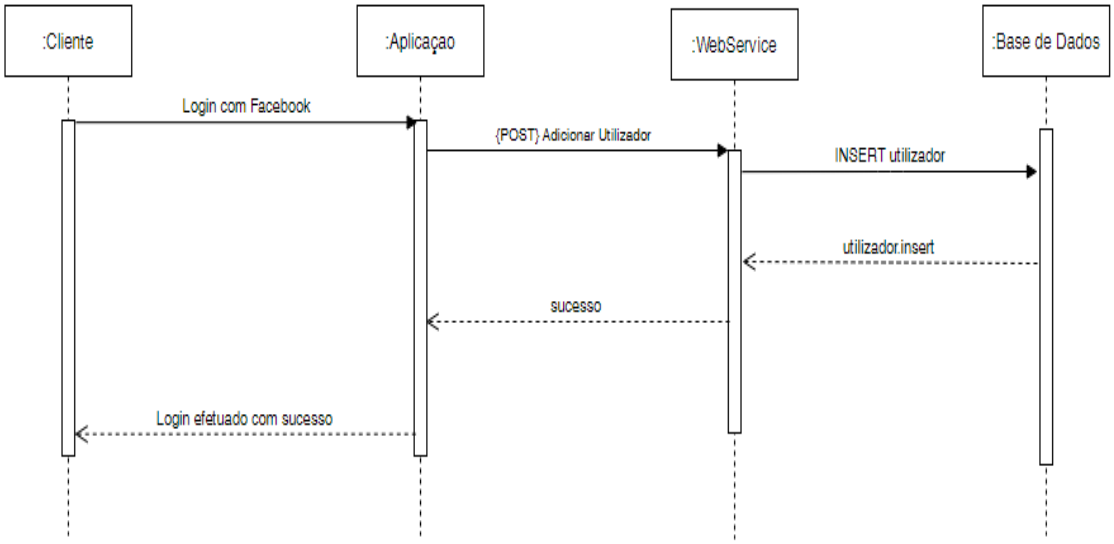


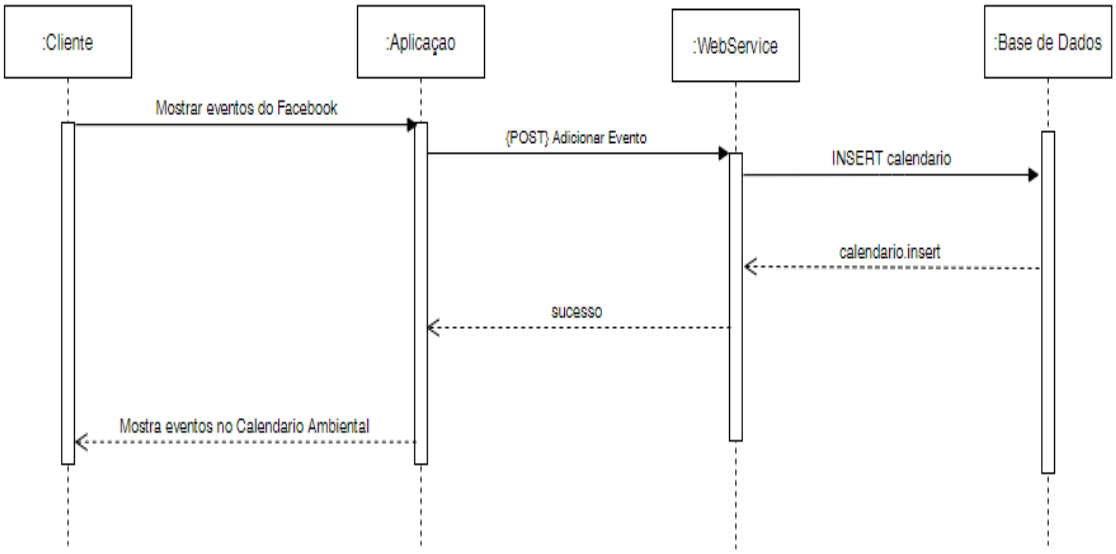

:Cliente
:Aplicaçao
:WebService
:Base de Dados
Adicionar Ecoponto
{POST} Adicionar Ecoponto
INSERT ecoponto
ecoponto.insert
sucesso
Mostrar ecoponto adicionado


:Cliente
:Aplicaçao
:WebService
:Base de Dados
Atualizar classificacao ecoponto
{POST} Atualizar Ecoponto
UPDATE rating
ecoponto.update
sucesso
Mostrar nova classificacao


:Cliente
:Aplicaçao
:WebService
:Base de Dados
Encontrar Ecoponto
{GET} Encontrar Top 5 Ecopontos
Classificacao TOP 5 WHERE cor=""
classificacao.select (cor="")
Retorna Ecoponto com menor distancia e maior classificacao
Encontra caminho até Ecoponto


:Cliente
:Aplicaçao
:WebService
:Base de Dados
Mostrar menu Ecoponto com classificação e último registo
{GET} Média Classificação e último registo
AVG (Classificacao) WHERE id Ecoponto=""
LAST (Classificacao) WHERE id Ecoponto=""
avg(classificacao).select (id="")
LAST(classificacao).select (id="")
Retorna média classificação e último registo de um Ecoponto
Mostra Ecoponto com média de classificação e último registo


:Cliente
:Aplicaçao
:WebService
:Base de Dados
Login com Facebook
{POST} Adicionar Utilizador
INSERT utilizador
utilizador.insert
sucesso
Login efetuado com sucesso


:Cliente
:Aplicaçao
:WebService
:Base de Dados
Mostrar eventos do Facebook
{POST} Adicionar Evento
INSERT calendario
calendario.insert
sucesso
Mostra eventos no Calendario Ambiental

Algoritmo MD5

A cifragem das *passwords* na aplicação é feita através do algoritmo MD5. Em seguida é representado o método de cifragem do algoritmo desenvolvido em *Java*.

```
public String encrypt(String unencryptedString) {
   String encryptedString = null;
   try {
      // Create MessageDigest instance for MD5
      MessageDigest md = MessageDigest.getInstance("MD5");
      //Add password bytes to digest
      md.update(unencryptedString.getBytes());
      //Get the hash's bytes
      byte[] bytes = md.digest();
      //This bytes[] has bytes in decimal format;
      //Convert it to hexadecimal format
      StringBuilder sb = new StringBuilder();
      for(int i=0; i< bytes.length ;i++)
      {
         sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,16).
         substring(1));
      }
      //Get complete hashed password in hex format
      encryptedString = sb.toString();
   }
   catch (NoSuchAlgorithmException e){e.printStackTrace();}
   return encryptedString;
}
```

Bibliografia

[1] J. N. Ferreira, "Quantos quilos de lixo produz um português por dia?," 2018. [Online]. Available: *HTTP*s://eco.sapo.pt/2018/08/07/quantos-quilos-de-lixo-produz-um-portugues-por-dia/.

[2] M. Gaio, "Concepção de uma aplicação móvel no âmbito da ecologia e *SUS*tentabilidade ambiental," 2014.

[3] "Invision," 2021. [Online]. Available: *HTTP*s://*WWW*.invisionapp.com/.

[4] "WasteApp," Quercus, 2019. [Online]. Available: *HTTP*s://*WWW*.wasteapp.pt/home.

[5] "Fundação Vodafone Portugal," Vodafone Portugal, 2021. [Online]. Available: *HTTP*s://*WWW*.vodafone.pt/a-vodafone/fundacao.*HTML*.

[6] "RecycleNation," Eri, 2020. [Online]. Available: *HTTP*s://recyclenation.com/.

[7] "ERI," 2021. [Online]. Available: *HTTP*s://eridirect.com/.

[8] "Wasteapp: esta app da Quercus ajuda a reciclar mais e melhor," 2019. [Online]. Available: *HTTP*s://tek.sapo.pt/mobile/apps/artigos/wasteapp-esta-app-da-quercus-ajuda-a-reciclar-mais-e-melhor.

[9] "Ecocentros," 2021. [Online]. Available: *HTTP*://*WWW*.valorsul.pt/pt/seccao/areas-de-negocio/triagem-de-materiais-reciclaveis/ecocentros.

[10] "Top 8 Innovative Recycling Apps That Make A Difference," 2021. [Online]. Available: *HTTP*s://APIumhub.com/tech-blog-barcelona/innovative-recycling-apps/.

[11] "Apache Tomcat," The Apache Software Foundation, 1999-2020. [Online]. Available: *HTTP*://tomcat.apache.org/.

[12] "Obtenha Informações sobre a Tecnologia *Java*," 2021. [Online]. Available: *HTTP*s://*WWW*.*Java*.com/pt-BR/about/.

[13] C. Draganova, Asynchronous *Java*Script Technology and *XML* (*AJAX*), 2007.

[14] "*Java* Script Object Notation," 2010. [Online]. Available: *HTTP*://*WWW*.*JSON*.org/.

[15] "JQuery," 2021. [Online]. Available: *HTTP*s://jquery.com/.

[16] "*Struts2*," 2000-2018. [Online]. Available: *HTTP*s://struts.apache.org/.

[17] "*Hibernate*," 2021. [Online]. Available: *HTTP*s://*Hibernate*.org/.

[18] "Flask," 2010. [Online]. Available: *HTTP*s://flask.palletsprojects.com/en/1.1.x/.

[19] "Bootstrap," 2011. [Online]. Available: *HTTP*s://getbootstrap.com/docs/4.0/getting-started/introduction/.

[20] "Gunicorn," 2021. [Online]. Available: *HTTP*s://gunicorn.org/.

[21] "Open Source ou Código Aberto: o que é, características e vantagens," 2020. [Online]. Available: *HTTP*s://fia.com.br/blog/open-source/.

[22] "Google Maps API," Google, 2020. [Online]. Available: *HTTP*s://developers.google.com/maps/documentation/*Java*script/geolocation.

[23] “Wikipedia,” 2021. [Online]. Available: *HTTP*s://pt.wikipedia.org/wiki/Criptografia.

[24] “Criptografia MD5,” 2007. [Online]. Available: *HTTP*s://*WWW*.devmedia.com.br/criptografia-md5/2944.

[25] “Software Engineering Institute,” Carnegie Mellon University, 2021. [Online]. Available: *HTTP*s://*WWW*.sei.cmu.edu/.

[26] V. Feng, “ResNet,” 2017. [Online]. Available: *HTTP*s://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[27] “Residual *CNN*s for Image Classification Tasks,” 2019. [Online]. Available: *HTTP*s://neurohive.io/en/popular-networks/resnet/.

[28] “Mailgun,” Mailgun Technologies, 2021. [Online]. Available: *HTTP*s://*WWW*.mailgun.com/.

[29] “Facebook *SDK* for *Java*Script,” 2021. [Online]. Available: *HTTP*s://developers.facebook.com/docs/*Java*script.

[30] G. Thung, “trashnet,” 2017. [Online]. Available: *HTTP*s://github.com/garythung/trashnet/blob/master/data/dataset-resized.zip.

[31] R. Atienza, Advanced Deep Learning with TensorFlow 2 and Keras, 2020.

[32] J. Jordan, “Setting the learning rate of your neural network.,” 2018. [Online]. Available: *HTTP*s://*WWW*.jeremyjordan.me/nn-learning-rate/.

[33] “Setting the learning rate of your neural network,” 2018. [Online]. Available: *HTTP*s://*WWW*.jeremyjordan.me/nn-learning-rate/.

[34] “Amazon S3,” 2021. [Online]. Available: *HTTP*s://my-hosted-temp.s3.us-east-2.amazonaws.com/trained_model_final.pkl.

[35] “How to build an image classifier for waste sorting,” 2019. [Online]. Available: *HTTP*s://towardsdatascience.com/how-to-build-an-image-classifier-for-waste-sorting-6d11d3c9c478.

[36] “Configurando *Hibernate* Passo a Passo,” 2013. [Online]. Available: *HTTP*s://*WWW*.devmedia.com.br/configurando-*Hibernate*-passo-a-passo/28652.

[37] “How To Create A Struts 2 *Web* Application,” 2018. [Online]. Available: *HTTP*s://struts.apache.org/getting-started/how-to-create-a-*Struts2*-*Web*-application.*HTML*.

[38] “Struts 2 Interceptor Example,” 2021. [Online]. Available: *HTTP*s://*WWW*.journaldev.com/2210/struts-2-interceptor-example.

[39] “fast.ai,” 2021. [Online]. Available: *HTTP*s://*WWW*.fast.ai/.

[40] “PyTorch,” 2021. [Online]. Available: *HTTP*s://pytorch.org/.

[41] “System Usability Scale (*SUS*),” 2021. [Online]. Available: *HTTP*s://*WWW*.usability.gov/how-to-and-tools/methods/system-usability-scale.*HTML*.

[42] L. Sharma, “How to Install Maven in Eclipse IDE,” 2014. [Online]. Available: *HTTP*s://*WWW*.toolsqa.com/*Java*/maven/how-to-install-maven-eclipse-ide/.

[43] “Heroku,” 2021. [Online]. Available: *HTTP*s://devcenter.heroku.com/articles/git.

[44] P. Lubbers, Overview of *HTML5*. In Pro *HTML5*, 2010.