INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Análise de Imagens na detecção de Automóveis Roubados

Diogo Filipe Rocha Rodrigues - a31814
Pedro Miguel da Silva Brito - a33386

Trabalho realizado sob a orientação de

Leonel Domingues Deusdado

Licenciatura em Engenharia Informática

2017

Análise de Imagens na detecção de Automóveis Roubados

Relatório da UC de Projeto

Licenciatura em Engenharia Informática

Escola Superior de Tecnologia e Gestão

Diogo Filipe Rocha Rodrigues - a31814

Pedro Miguel da Silva Brito - a33386

2017

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Dedicatória

Este trabalho é dedicado à família, que sempre apoiaram nos estudos. Dedicado também aos amigos e coordenador pela sua colaboração e apoio.

Agradecimentos

Este trabalho não ficaria completo sem agradecer a todos os que nos ajudaram a concretizá-lo. Em primeiro lugar, queríamos agradecer ao nosso orientador, Professor Leonel Domingues Deusdado, pela sua ajuda, orientação, paciência e simpatia. Gostaríamos de agradecer aos nossos amigos que de certa forma foram dando o seu contributo para a realização do projeto. A todos o nosso obrigado.

Resumo

A análise e processamento de imagens é uma tecnologia com um grande poder de progressão, no qual existe diversas áreas em que pode ser aplicada, desde a segurança, à medicina, passando pelo nosso próprio entretenimento. O interesse da área, aliado à necessidade de realizar a unidade curricular de Projeto, é uma mais valia para que o mesmo seja realizado com interesse e boa vontade.

A aplicação de forma resumida, permite fazendo o uso da análise e tratamento de imagens extrair matrículas de fotografias tiradas a automóveis, e alguns dados considerados relevantes neste projeto como datas, horas e localizações de onde as fotografias foram tiradas.

Fazendo uso destes dados, é possível criar uma base de dados de veículos que foram roubados, possibilitando que esta aplicação se transforme em uma ferramenta que ajude a recuperar os mesmos, podendo ser utilizada de forma proveitosa pelas forças de segurança.

Abstract

The analysis and processing of images is a technology with a great power of progression, in which there are several areas in which it can be applied, from safety, to medicine, to our own entertainment.

The interest of the area, together with the need to carry out the Project curricular unit, is an added value for it to be carried out with interest and goodwill.

The application, in a summary form, allows the use of image analysis and treatment to extract license plates from photographs taken of the vehicles and some data considered relevant in this project, such as dates, times and locations from which the photographs were taken.

Using this data, it is possible to create a database of vehicles that have been stolen, allowing this application to become a tool that helps to recover them and can be used in a useful way by security forces.

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

ALPR Automatic License Plate Recognition. 4, 6, 9, 14

ER Entidade Relacionamento. 13, 15

OCR Optical Character Recognition. 5

Capítulo 1

Introdução

O projeto realizado, pode ser considerado como uma ferramenta de segurança, ou como um meio de entretenimento, sendo inspirado no jogo Pokémon GO [1]. Permitindo a interação dos jogadores com o meio que os rodeia, fazendo uso de recursos de um smartphone.

O objetivo da aplicação é que os utilizadores façam uso da câmera fotográfica do smartphone e procurem carros que tenham sido roubados, anunciados previamente por parte dos utilizadores lesados registados na mesma aplicação.

A matrícula do automóvel roubado é desconhecida totalmente por parte dos utilizadores que o procuram, sendo apenas conhecida uma descrição do mesmo, presente no anúncio do automóvel lesado.

De forma a manter o interesse no uso da aplicação, o objetivo é criar um sistema de ranking e de recompensas monetárias por cada veículo encontrado, sendo que essas recompensas variam de acordo com o quanto o utilizador lesado está disposto a oferecer.

A base de dados da aplicação contendo todos os dados recolhidos por parte dos utilizadores que procuram veículos, vai permitir que automóveis roubados sejam detetados logo após a publicação do anúncio na aplicação, usando apenas o histórico

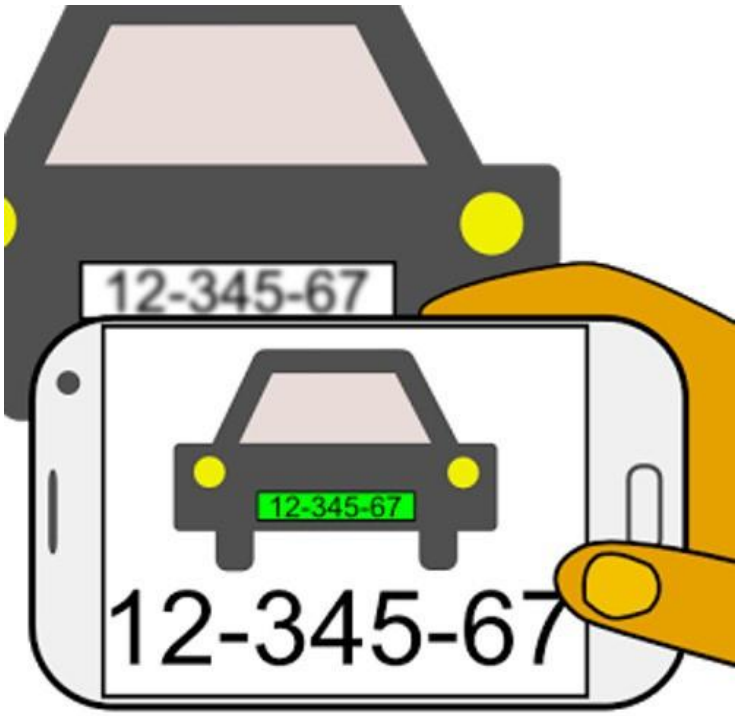

Figura 1.1: Logotipo [2]

Capítulo 2

Recursos Utilizados

2.1 OpenALPR

ALPR (Automatic License Plate Recognition) é uma biblioteca de código livre, originalmente escrita C++, que pode ser também implementada em Java, CSharp e Python. O objetivo desta biblioteca, sendo a base deste projeto, é a análise de uma imagem, de forma a verificar se existe uma matrícula, e proceder à extração da mesma.

2.1.1 Funcionamento do ALPR

Depois de obtida a fotografia do automóvel, contendo a matrícula do mesmo, é necessário isolar a área correspondente da matrícula, usando um algoritmo que deteta os seus contornos.

Figura 2.1: Recorte da matrícula

Obtida a área corresponde da matrícula, é necessário aplicar uma rotação e inclinação, de forma a colocar a matrícula horizontalmente. Este processo é necessário porque as fotografias tiradas ao automóvel podem ser de diversos ângulos.

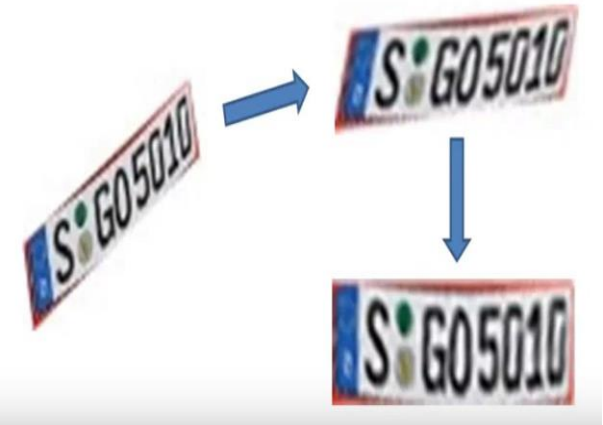

Figura 2.2: Posicionamento da matrícula

O próximo passo que o Automatic License Plate Recognition (ALPR) realiza, é ajustar o brilho e o contraste da área recortada da matrícula, de forma a obter uma imagem com alto contraste, o que facilita na deteção dos caracteres da matrícula.

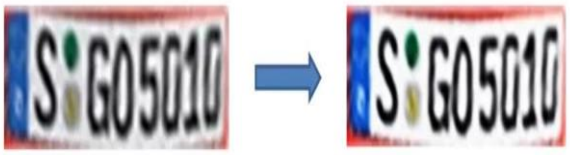

Figura 2.3: Matricula com brilho e contraste ajustado

Detetadas as áreas correspondentes aos caracteres, é necessário segmentar todos os caracteres da matrícula, isto significa, dividir a matrícula em várias partes, cada parte contendo o seu número ou letra. Isto é necessário para que cada segmento da matrícula possa ser preenchido no Optical Character Recognition (OCR) (Optical Character Recognition)

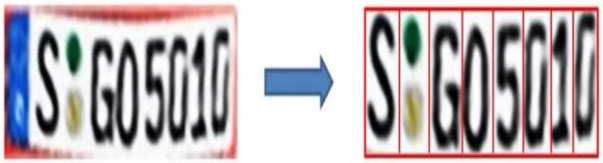

Figura 2.4: Segmentar a matrícula

O OCR em português (Reconhecimento Ótico de Caracteres) irá tratar um segmento da matrícula de cada vez, colocando os resultados numa string, sendo o valor dessa string o resultado final da extração da matrícula obtida a partir da imagem.

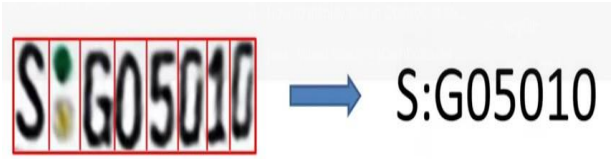

Figura 2.5: Resultado da extração

2.1.2 Desafios do ALPR

O ALPR embora seja muito usado em diversas áreas como parques de estacionamento, veículos da polícia entre outras, apresenta falhas. A falha do algoritmo na leitura da matrícula, pode ter origem em diversas varáveis como:

- Distância da câmara no momento em que a fotografia do automóvel foi tirada.

- Pouca luz, impedindo o funcionamento correto do algoritmo.

- Fotografia com pouca resolução, dificultando a deteção da área da matrícula na fotografia, ou caso a área da matrícula tenha sido encontrada, pode ocorrer dificuldades em detetar os caracteres da matrícula, podendo levar a valores incorretos no resultado extraído.

- Processamento demorado das imagens. Esse atraso pode ser muito penoso para os utilizadores, quando se trata de grandes quantidades de imagens. Podendo ser necessário máquinas adicionais para o processo.

- A grande quantidade de matrículas existente de diferentes tamanhos, cores, com imagens de fundo e tipos de letra distintos, pode ser um entrave na utilização do algoritmo. Por norma na criação do algoritmo deve se ter em conta o país para o qual vai ser desenvolvido, isto permite que o algoritmo saiba as combinações de matrícula existentes num determinado país, garantido que numa certa posição da matrícula vão aparecer só letras, e em uma outra só números, garantindo um maior grau de confiança.

2.1.3 Implementação do OpenALPR

A implementação e configuração da biblioteca OpenALPR no computador, foi feita usando o seguinte tutorial do GitHub [3]

2.1.4 Execução do OpenALPR na shell

O OpenALPR permite ser executado via linha de comandos.

O comando que executa a biblioteca é composto por diversos parâmetros, que permitem definir uma configuração para a imagem que vai ser processada. Entre esses parâmetros estão: o caminho onde a imagem está guardada, o país identificador das matrículas que o algoritmo vai analisar, e um número de resultados que pretendemos obter com o respetivo grau de confiança em relação à imagem analisada, como mostra na figura 2.6.

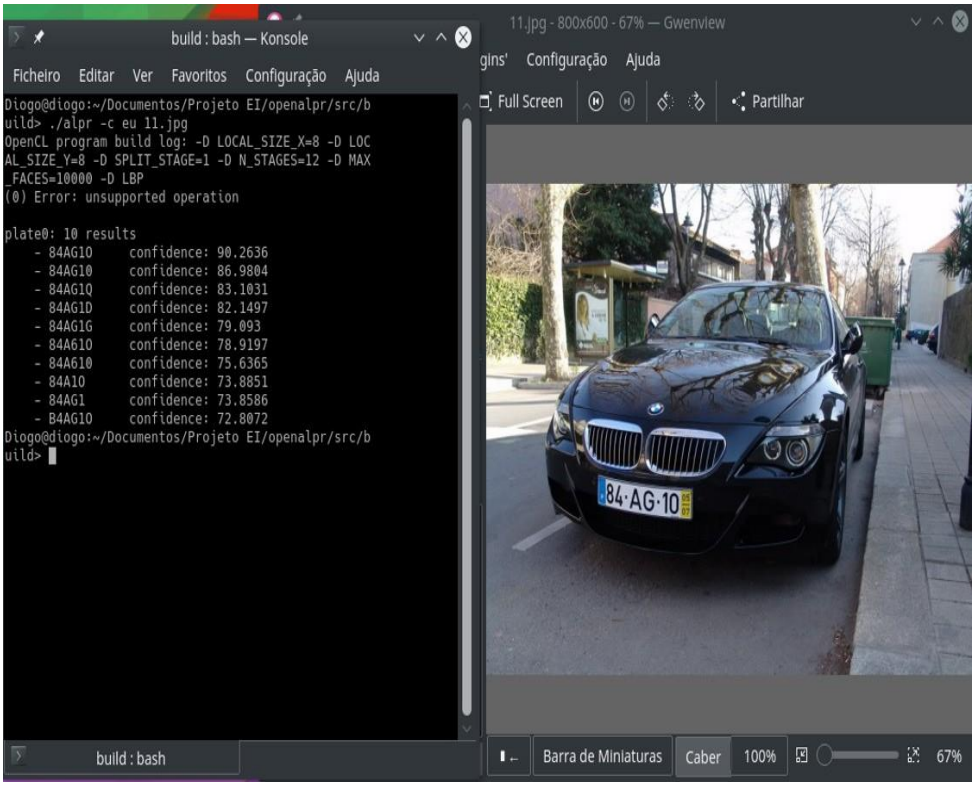

Figura 2.6: Execução OpenALPR

2.1.5 Fiabilidade daAplicação

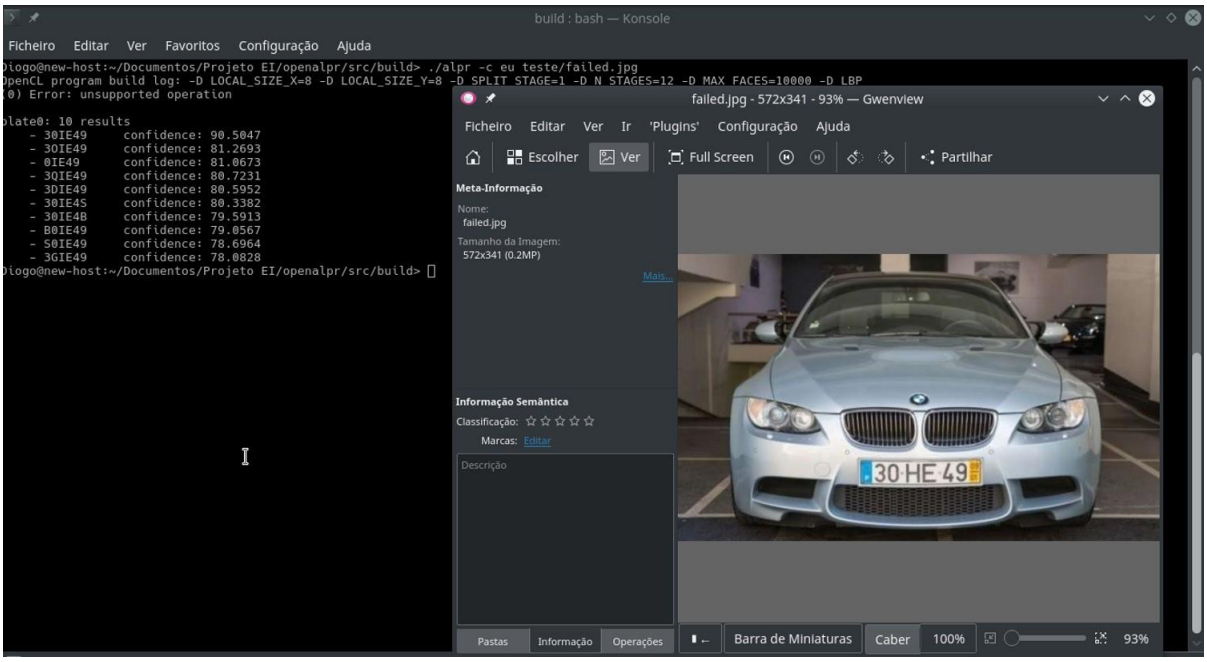

Figura 2.7: Leitura Falhada

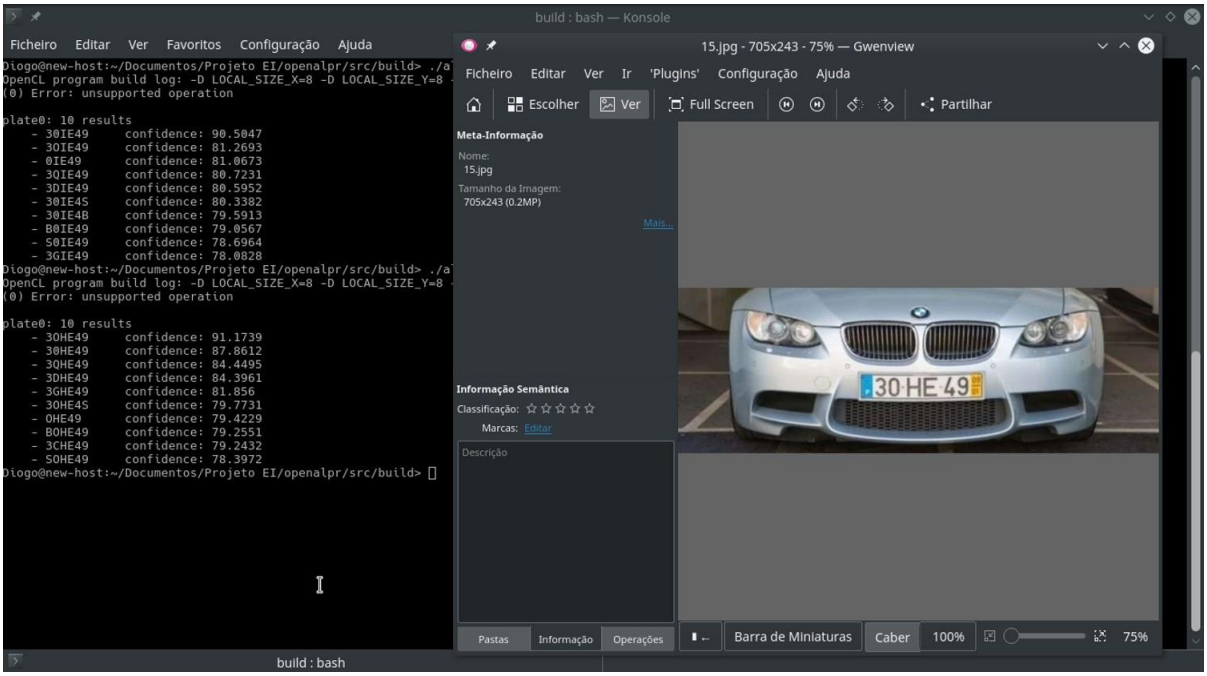

Figura 2.8: Imagem Editada

Como explicado anteriormente o algoritmo ALPR apresenta vários desafios, que podem originar várias falhas oriundas de múltiplas causas, pondo em causa a veracidade de toda a aplicação.

Imagine-se o caso de um utilizador em busca de carros roubados, enviar fotos para o servidor para que este proceda à extração das mesmas, e consequente comparação desse valor extraído com as matrículas presentes na base de dados, referente a veículos roubados. Se esse valor extraído pelo algoritmo for diferente do da foto, pode levar a casos de falsos alertas e recompensas atribuídas indevidamente.

De forma a contornar o problema, um dos planos defendidos é fazer uso dos graus de confiança que o algoritmo fornece por cada foto analisada. Isto permite, com base num estudo realizado verificar se é vantajoso ou não, definir um limite mínimo de grau de confiança, para que uma foto seja aceite pelo servidor, e registada na base de dados.

Caso a foto enviada por um dado utilizador não obtenha esse limite mínimo de grau de confiança, esse mesmo utilizador é informado que a sua foto não foi validada, devido à sua fraca qualidade. Pedindo ao utilizado que volte a tentar tirar uma nova foto.

Num teste efetuado, foram analisadas 20 fotos em que o posicionamento da câmara e luminosidade são favoráveis para que o algoritmo funcione corretamente. Dessas 20 fotos definiu-se um limite mínimo em que o grau de confiança do valor extraído da foto deve ser atingido.

Durante a realização do teste, das 20 fotos uma deu um falso positivo, ou seja, a foto atingiu um resultado de confiança superior a 90 por cento embora o seu valor fosse incorreto. O valor correto extraído foi alcançado através de uma nova foto, desta vez mais próxima da matrícula do automóvel.

Na realização do teste foi possível também verificar que detalhes como a cor do veiculo e o ângulo da foto podem dificultar o algoritmo a encontrar a área da matrícula.

Resultados obtidos:

Com base nos resultados obtidos, pode-se que concluir que limitar o grau de confiança de forma a verificar se a foto pode ser validada, apenas vai solucionar o problema de forma parcial, existindo sempre a probabilidade de o resultado extraído da foto ser um

Grau de Conftança	Coincide	Não Coincide
>= 90 %	16	1
< 90 %	2	1

Tabela 2.1: Resultados do teste

falso positivo.

Se a solução passar por aumentar esse limite mínimo do grau de confiança, pode tornar a aplicação obsoleta e pouco atrativa.

A única maneira de contornar o problema é a confirmação por parte do utilizador que enviou as fotos, do resultado extraído.

Capítulo 3

Linguagens e Métodos Adotados

Neste capitulo é possível dar a conhecer as linguagens que foram utilizadas durante a realização do projeto, assim como os métodos adotados para componentes do projeto ainda não concluídos.

Mais do que dar a conhecer as linguagens utilizadas, pretende-se também indicar um conjunto de razões que motivaram a tal escolha.

3.1 Servidor

Java foi a linguagem usada na construção de todos os componentes que compõe o servidor da aplicação.

3.1.1 Motivos para a utilização do Java

- Um maior conhecimento comparando com outras linguagens de programação, assim como sua portabilidade, ou seja, capacidade de o Java conseguir correr em vários sistemas operativos como Linux, Windows, MacOS entre outros.
- Simplicidade na linguagem, uma vez que é derivada do c e c++ tornando-a em um ambiente mais familiarizado.
- Como é uma das linguagens mais utilizadas, tem a vantagem de ter uma grande comunidade, onde é possível obter ajuda, e materiais de estudo.
- Conhecimento mais aprofundado sobre sistemas distribuídos usando a linguagem Java, essencial para a realização de projeto.

3.2 Cliente

Android A aplicação foi desenvolvida para o sistema Android, isto porque existe um interesse de adquirir conhecimento com o seu desenvolvimento, assim como uma maior facilidade no uso do Java por parte do Android.

Capítulo 4

Arquitetura do Sistema

A arquitetura do sistema foi construida utilizando alguns modelos técnicos como Entidade Relacionamento (ER) e casos de uso, que permitiram o planeamento e consequente desenvolvimento do projeto.

Sendo assim é descrito neste capitulo, a arquitetura de todos os componentes do sistema, e a forma como eles se relacionam entre si.

4.1 Arquitetura da Base de Dados

A aplicação irá girar em torno da base de dados, realizando constantemente operações de inserção e consulta de dados, tornando-se assim a base do projeto um dos elementos mais importantes.

Apache Derby base de dados implementada inteiramente em Java, que foi incorporada no projeto do servidor, permitindo um maior controlo e portabilidade da mesma, evitando assim a instalação prévia de algum tipo de base de dados na máquina servidor.

4.1.1 Descrição da Base de Dados

A base de dados implementada no projeto, deve ser capaz de registar novos utilizados, assim como permitir que eles efetuem um inicio de sessão na aplicação. Uma vez com a

sessão iniciada na aplicação, o cliente deve ser capaz de fazer uma consulta de anúncios presentes na base de dados, anúncios esses referentes a carros roubados, publicados por outros utilizados com os mesmos privilégios.

Prevendo um decréscimo na performance da base de dados, devido ao uso de muitas imagens, definiu-se que para qualquer campo de imagem ia resultar numa tabela própria para essa imagens, permitindo assim que as imagens fossem consultadas apenas quando necessário.

A tabela **Client**, responsável por guardar os dados da conta do cliente, está associada a uma tabela **Vehicle**. Essa tabela permite guardar dados referentes a um ou mais veículos, que sejam propriedade de um utilizador. Por sua vez e de forma opcional, a tabela **Vehicle** está associada a uma tabela **PhotographyVehicle**, responsável por guardar múltiplas fotos de um automóvel.

O objetivo de se associar os veículos ao utilizador, é permitir que caso seja necessário criar um anuncio, esse anúncio seja criado de forma rápida, contendo todas as informações referentes ao veiculo, assim como as fotos do próprio veiculo.

Cada envio de fotos, é registado pela tabela **Send**, que contem datas de quando o pedido chegou ao servidor, e datas de quando ele foi concluído

Depois de registado o envio de fotos, o servidor ALPR recebe essas fotos, sendo ele responsável pela extração de todas as informações da foto, como matrícula, hora, data e geolocalização, adicionando de seguida essas informações na tabela **PhotographyData**, existente na base de dados.

A tabela **Advertising**, vai receber consultas de forma constante, por parte dos utilizadores que pretendem ver os anúncios existentes, e por parte do servidor que irá efetuar uma consulta sempre que extrair dados de uma foto enviada para o servidor, comparando a matrícula extraída de uma foto, com as matrículas presentes na tabela **Advertising**.

Se o resultado dessa consulta por parte do servidor for positiva, ou seja, se essa matrícula existir na tabela **Advertising** e se o anúncio tiver ativo, a tabela **Payment** vai receber o **id** desse anúncio, possibilitando que o pagamento seja efetuado entre quem publicou o anúncio, e quem fez um envio de fotos para o servidor.

4.1.2 Modelo ER

Modelo ER da base de dados que posteriormente permitiu a sua criação.

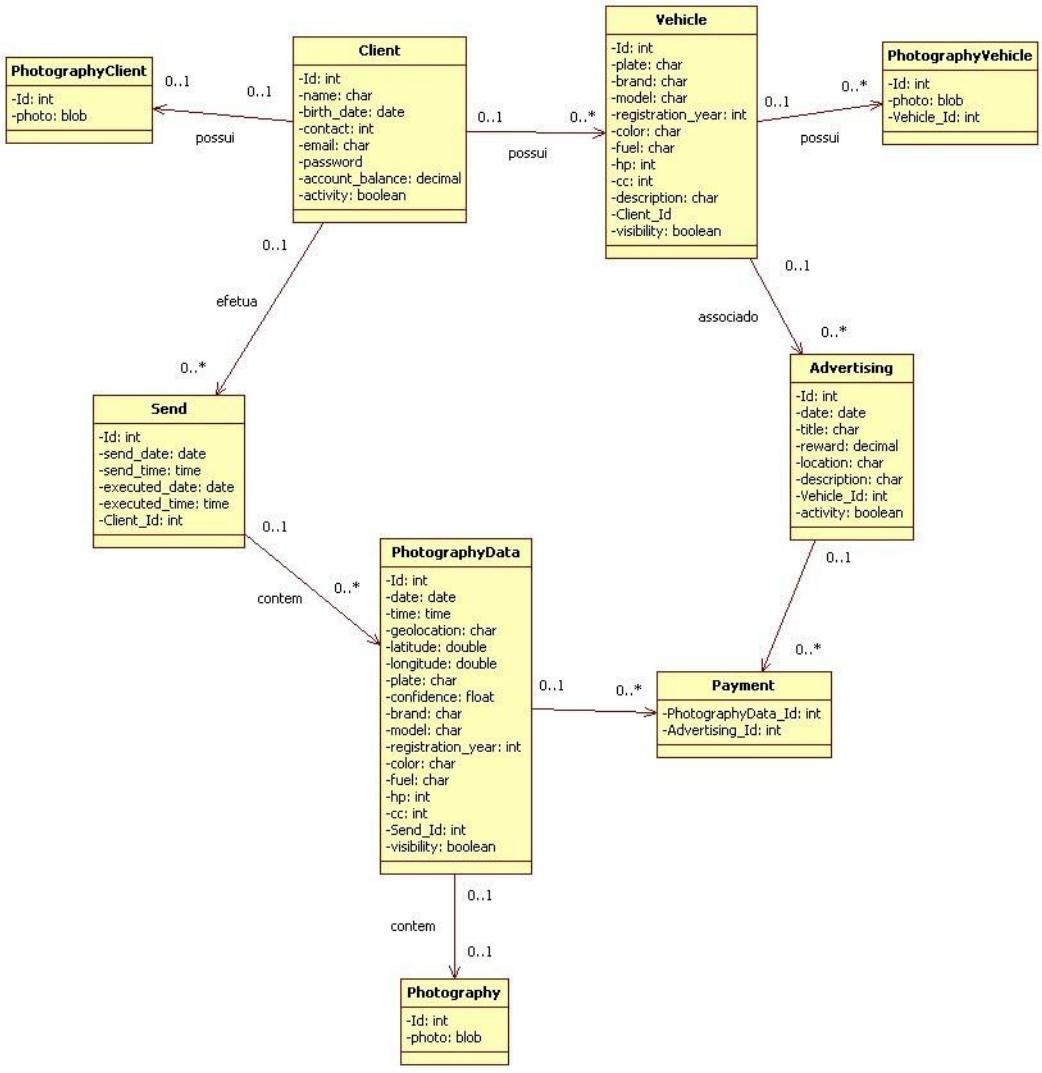

Figura 4.1: Modelo ER

4.1.3 Modelo MySQL Workbench

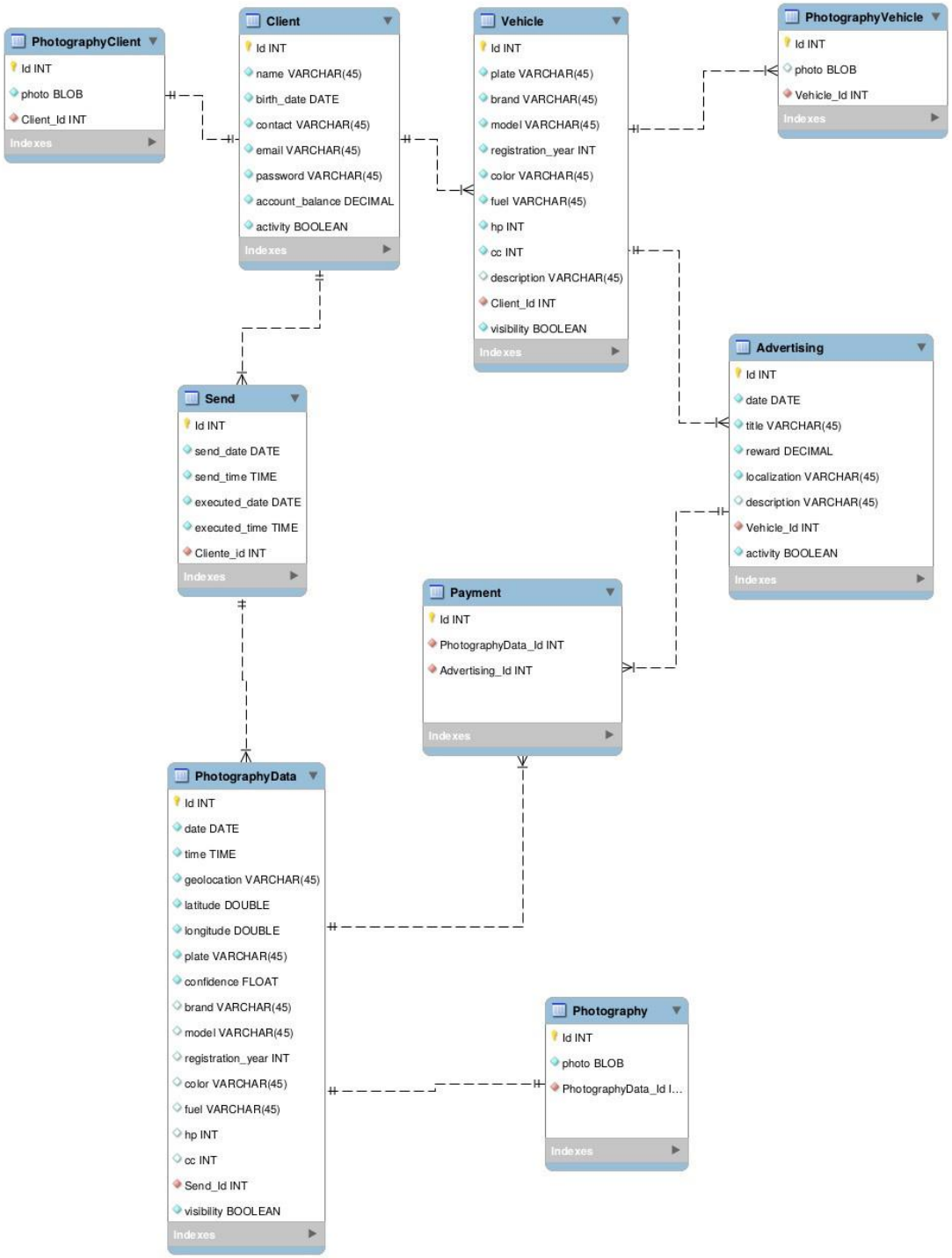

Figura 4.2: Modelo MySQL Workbench

4.2 Arquitetura do Sistema

Foi criado um modelo simplista da arquitetura do sistema, de forma a facilitar a descrição e o entendimento do sistema em si, e como os seus componentes se relacionam.

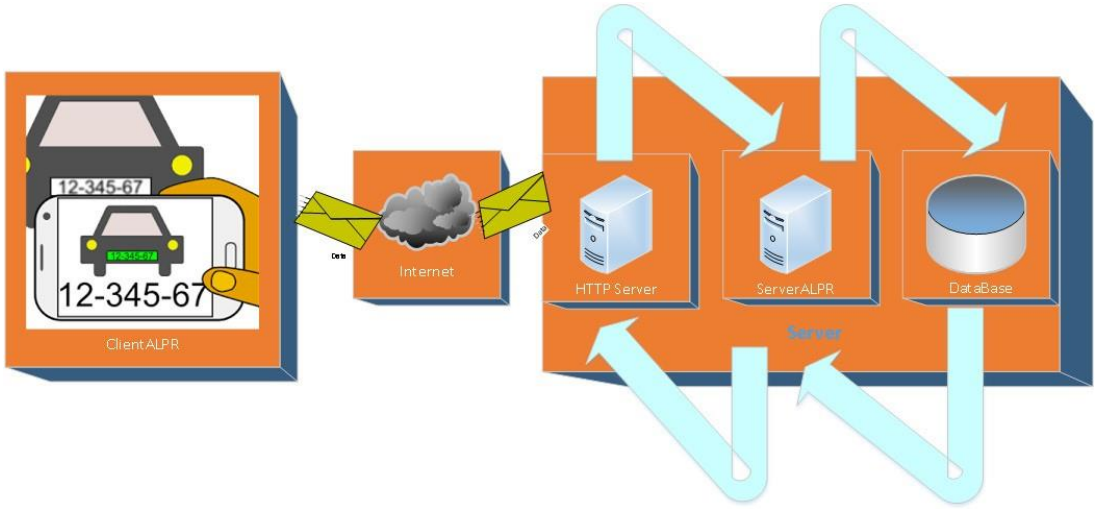

Figura 4.3: Modelo do Sistema

4.2.1 Descrição do Sistema

O sistema está assente em duas bases, uma que é referente ao servidor e todos os componentes que lhe dizem respeito, e a outra que é a aplicação.

Cliente

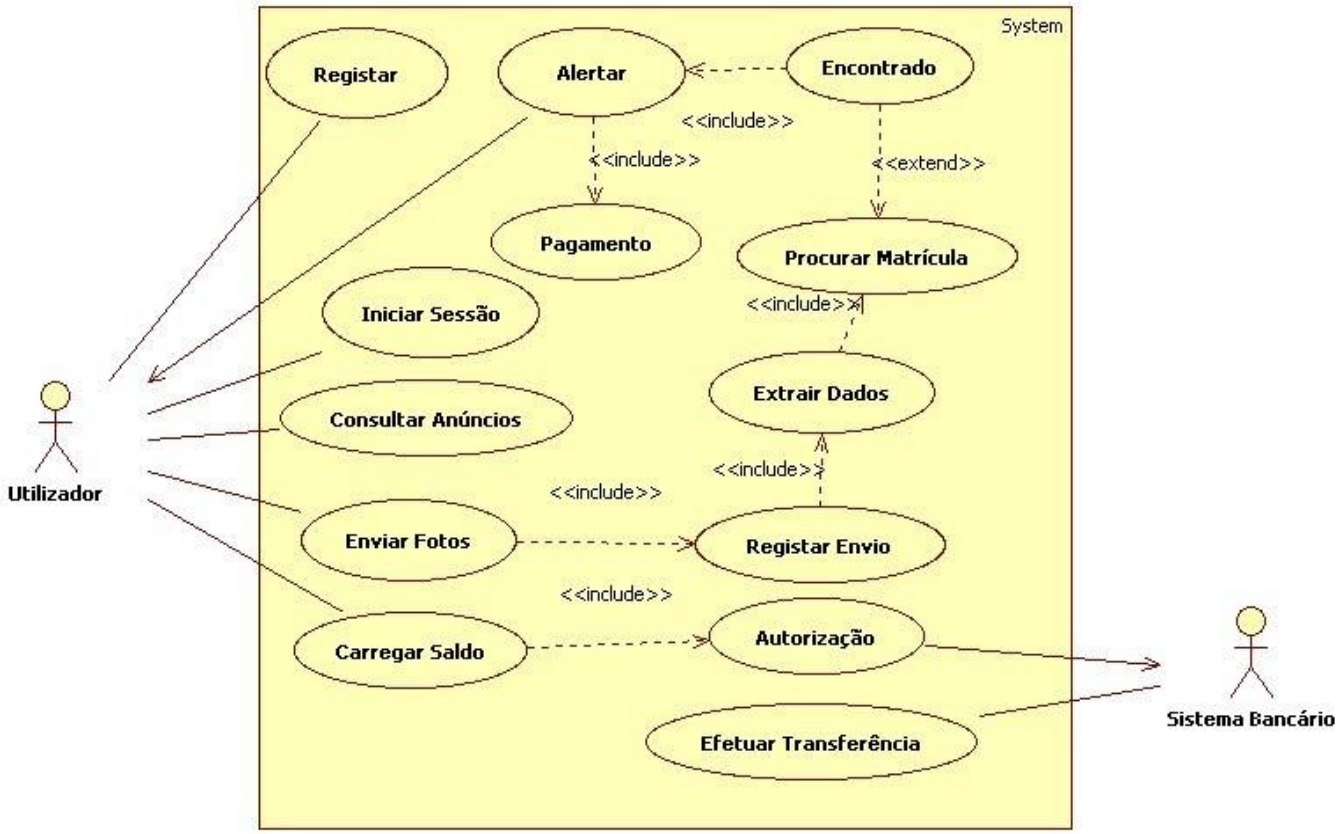

Figura 4.4: Modelo Casos de Uso

Aplicação A aplicação deve ser capaz de comunicar com o servidor através de um **WebService restful**. Para que a comunicação entre servidor e cliente seja possível, é necessário recorrer ao **JSON**, de forma a converter os objetos java para objeto json e vice-versa, sendo necessário a existencia de classes com os mesmos atributos dos dois lados (cliente e servidor), para que as conversões sejam feitas corretamente.

Funcionalidades fornecidas pela aplicação móvel:

- Registo de novos utilizadores.
- Início de sessão em qualquer momento, por parte dos utilizadores registados.
- Uma vez com início de sessão, é possível registar os veículos dos quais o utilizador é proprietário.
- Consulta de todos os anúncios disponíveis, referentes a veículos furtados previamente publicados por outros utilizadores registados. Podendo ser efetuadas pesquisas por zona, caso se pretenda.
- Possibilidade de criar um anúncio de forma simples, referenciando o automóvel furtado, e oferecendo uma recompensa monetária à sua escolha, para quem o encontrar primeiro.
- Carregar dinheiro na conta de utilizador. Dinheiro esse que vai ser utilizado para o pagamento de recompensas.
- Envio de fotos, por parte de quem procura automóveis roubados.

  A aplicação deve ser capaz de enviar propriedades referentes à câmera fotográfica da foto que tirou para o servidor. É também necessário que em cada foto, seja escrito a latitude e a longitude, de forma a contornar a não ativação do geotag, por parte do utilizador.

Servidor A correta implementação do servidor é essencial para que a aplicação funcione corretamente.

É necessário a existência de uma arquitetura que forneça uma sincronização total entre todos os componentes do servidor, isto porque o servidor vai funcionar em ciclos constantes, repetindo de forma continua as mesmas tarefas e instruções.

Sendo a análise de imagem a principal característica da aplicação, e sendo essa análise feita no lado do servidor, é natural a existência de uma grande quantidade de dados enviada entre o cliente e o servidor, podendo tornar a aplicação pouco viável devido ao tempo demorado gasto no tratamento de dados. Logo é necessário que o servidor adote políticas de eficiência, de forma a que o tempo gasto no tratamento de dados seja menor, e impedir que não existam utilizadores beneficiados e outros prejudicados em relação à prioridade que o servidor dá no tratamento dos seus dados.

De forma a tornar o processo realizado pelo servidor mais eficaz, é necessário implementar na sua arquitetura, conceitos de multi-threading e de sincronização no acesso a recursos críticos.

Existem dois tipos de utilizadores que fazem uso dos recursos do servidor, os utilizadores cujo o seu veiculo foi roubado, e tentam publicar anúncios na aplicação, e os utilizadores que vão em busca de carros roubados, e enviam constantemente fotos para o servidor poder tratar.

Capítulo 5

Funcionamento do Servidor

O objetivo deste capítulo é descrever o funcionamento do servidor, assim como a sua interação com outros componentes do sistema.

5.1 Funcionalidades

5.1.1 Serviço Carro

Este serviço implementado no projeto, permite identificar o veículo presente em cada foto, fornecendo ao utilizador uma descrição detalhada da marca, modelo, cor, combustível, etc...

O serviço consiste em realizar extrações massivas de dados de uma página web, através de pedidos GET e POST.

A plataforma web utilizada para a extração de dados foi o Sapo Venda Já, serviço que fornece as características de um automóvel, sendo necessário apenas fornecer a matrícula.

A possibilidade de registar um automóvel na aplicação utilizando apenas fotos, foi um dos motivos para o desenvolvimento deste serviço.

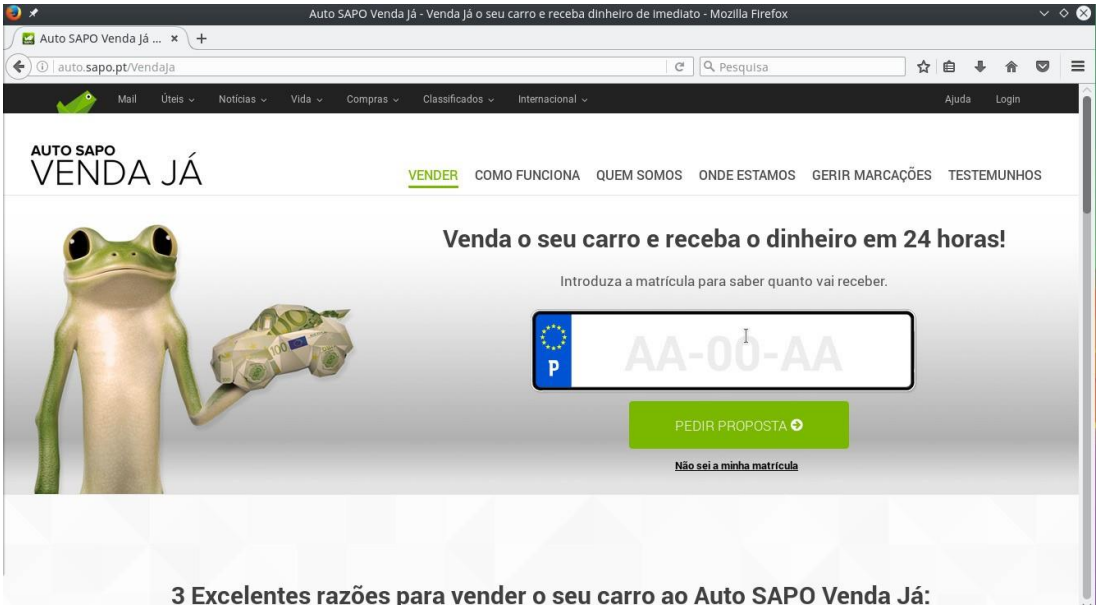

Figura 5.1: Plataforma Web - Sapo Venda Já [4]

```
193.107.85.56

Sending 'GET' request to URL : http://auto.sapo.pt/VendaJa/
Response Code : 200
Extracting form's data...

Sending 'POST' request to URL : http://auto.sapo.pt/VendaJa/
Post parameters : __RequestVerificationToken=xWUssLz1EoUVGCH6rRpZmxyr
Response Code : 200

Sending 'GET' request to URL : http://auto.sapo.pt/VendaJa/Vender
Response Code : 200
```

Figura 5.2: POST e GET - Sapo Venda Já

5.1.2 Serviços de Redes

Serviço que surgiu da necessidade de contornar os mecanismos de bloqueio da plataforma Sapo Venda Já, que ocorrem devido ao acesso constante por parte do servidor ALPR.

Como forma de contornar os constantes bloqueios devido ao elevado número de acessos, a solução passa por realizar todas as comunicações através de servidores proxy da rede TOR. Isto possibilita que a cada 5 acessos a esta plataforma um novo proxy seja utilizado para a comunicação, dando assim a noção de que clientes estão diferentes estão a aceder aos seus serviços.

5.1.3 Serviço de Geolocalização

Como foi referido anteriormente, é necessário que a aplicação seja capaz de escrever a latitude e a longitude em cada fotografia que envia para o servidor, sendo esta API responsável por converter a latitude e longitude em endereços de morada compreensíveis para o utilizador.

5.1.4 Distância da Matrícula

Funcionalidade que não estava prevista no plano original do servidor, e que permite calcular a distância em metros ou centímetros, da foto tirada para a matrícula do automóvel.

Calculo que só é possível fazendo uso dos parâmetros da câmera que tirou a fotografia, parâmetros esses: Focal Length, Sensor Width e Sensor Height. No calculo da distância, estamos a partir do principio que nenhum zoom foi feito na câmera fotográfica, o que pode causar valores errados. É importante referir que a distância obtida não é exatamente correta, existindo um desfasamento de 1.5cm a 2cm, tudo dependendo de como a área da matrícula foi reconhecida, nunca sendo reconhecida exatamente nas suas extremidades.

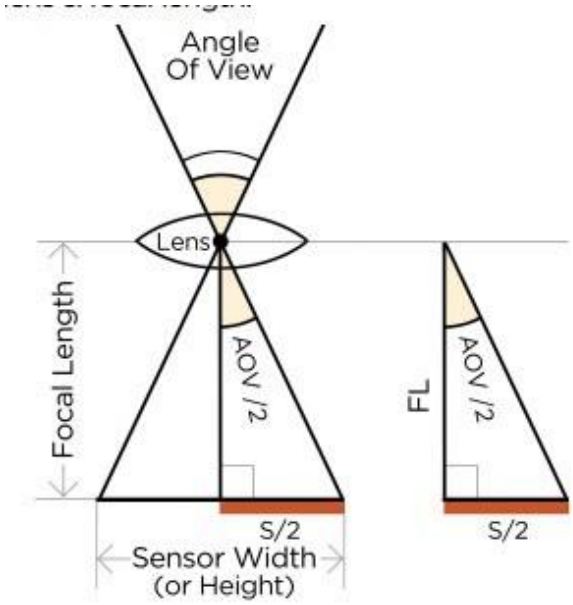

Figura 5.3: Arquitetura da câmera fotográfica

$$\text{Distance to object}(mm) = \frac{f(mm) \times \text{real height}(mm) \times \text{image height}(pixels)}{\text{object height}(pixels) \times \text{sensor height}(mm)}$$

Figura 5.4: Fórmula de calculo

5.1.5 Matrícula ocultada

Fotografias enviadas para o servidor, que tenham como objetivo o registo de um novo automóvel, podem ver a área correspondente à sua matrícula, ocultada de forma automática. Funcionalidade importante para que nos anúncios publicados, a matrícula do automóvel esteja sempre oculta para quem o procura.

5.1.6 Restful WebService

Implementam todos os métodos necessários para serem acedidos de forma remota, assim como um mecanismo de segurança para prevenir que pessoas não autorizadas façam uso deles.

Cada pedido feito pela aplicação ao servidor, irá gerar um token único, que ficará do conhecimento do servidor e do cliente que fez o pedido, permitindo que os métodos só sejam acedidos por quem tem o token autorizado pelo servidor.

Figura 5.5: Ocultação de matrícula

Objetos JSON são transferidos entre cliente e servidor sendo necessária em ambas as partes a sua conversão para objetos Java. Para que essa conversão seja feita corretamente é importante garantir que o objeto Java que vai ser instanciado apartir do JSON, tenha os mesmos atributos do objeto Java do outro lado.

5.2 Ciclo doServidor

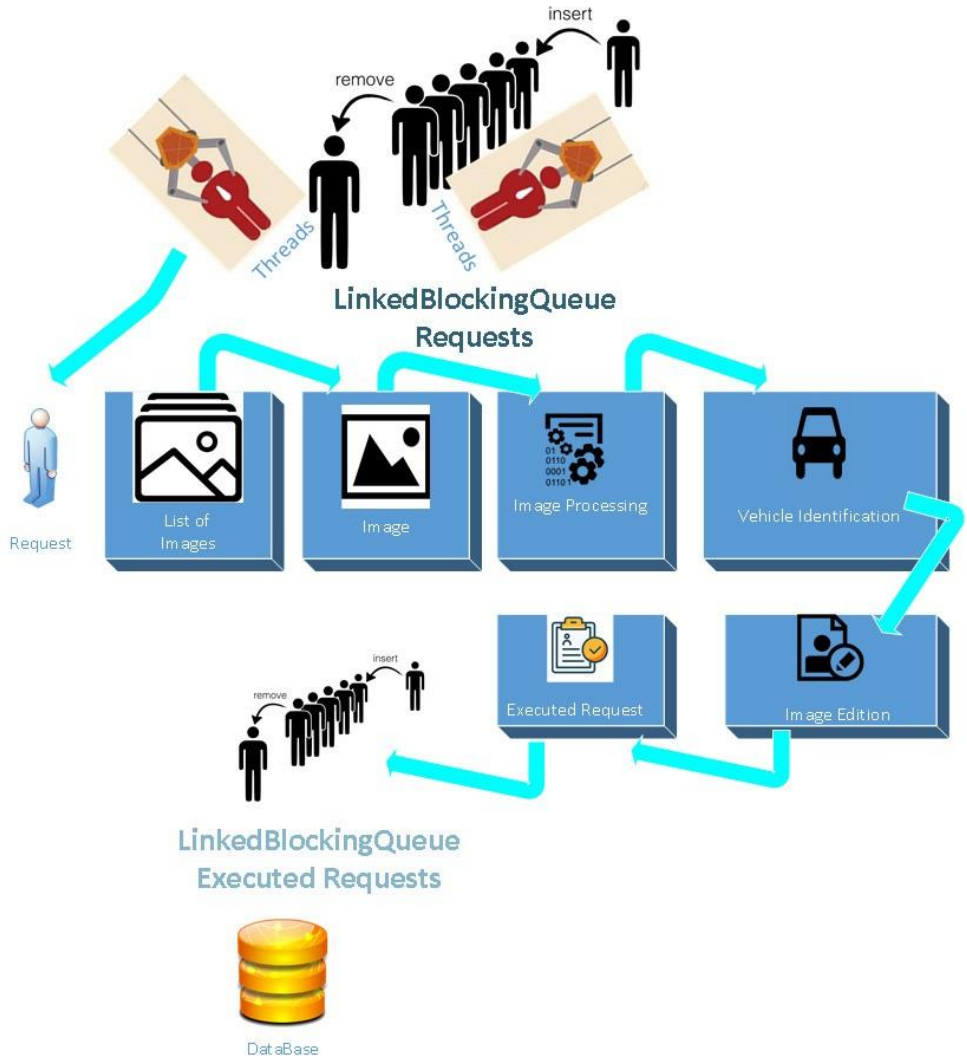

Figura 5.6: Arquitetura Detalhada do Servidor

A Figura 5.5 mostra uma arquitetura detalhada do servidor.

O servidor é composto por 2 filas**(LinkedBlockingQueue)**, uma para inserir pedidos feitos pelos clientes, e a outra para inserir na base de dados os pedidos executados.

Cada pedido que chega é identificado pelo id do cliente que fez o pedido, assim como os dados desse pedido, ou seja, as fotos enviadas e os parâmetros da câmera de cada foto. É também necessário que o pedido seja identificado quanto ao tipo de pedido, que pode variar em duas opções.

Caso o pedido seja do tipo 0, significa que as fotos enviadas servem para registar novos veículos na conta de cliente que os enviou, se for do tipo 1, significa que as fotos enviadas são para verificar se existe algum veículo roubado.

De forma a acelerar a execução de pedidos por parte do servidor, threads são criadas consoante os recursos do CPU. Essas threads pegam nos pedidos que chegam na lista e começam a executar-los

É necessário garantir que certas partes do código sejam sincronizadas, de forma a garantir que apenas uma das threads o está a executar, isto porque a biblioteca que realiza o processamento das matrículas não está preparada para realizar multi-processamento.

O uso de múltiplas threads só é vantajoso se o número de pedidos no servidor assim o justificar, isto porque criar novas threads pode ser custoso a nível de tempo.

Como mostra na Figura 5.5 um pedido pode ter várias fotos, em que cada uma dessas fotos vai ser processada.

O primeiro passo realizado em cada uma das fotos, é a extração dos meta dados da foto, de forma a verificar se essa foto contem ou não coordenadas de localização, rejeitando as que não possuírem.

Caso as coordenadas estejam presentes, a biblioteca **ALPR** encarrega-se de encontrar a área válida da matrícula na fotografia, e proceder á sua identificação, rejeitando qualquer foto cuja área não tenha sido encontrada. Obtida a matrícula da fotografia, assim como as coordenadas correspondentes à área da matrícula na fotografia, o serviço carro explicado anteriormente, encarrega-se de identificar o carro identificado pela matrícula extraída.

Se o pedido for do tipo 0, é necessário que área correspondente à matrícula na fotografia seja ocultada, caso seja 1, é calculada a distância até à matrícula, funcionalidade que foi

explicada anteriormente.

O processamento termina quando os resultados do pedido tratado são inseridos na lista de pedidos executados, para uma posterior inserção na base de dados.

Capítulo 6

Implementação da Aplicação

6.1.1 MainActivity

Esta classe é o main da aplicação, onde é criado o latyout inicial da aplicação. A classe MainActivity implementa NavigationView.OnNavigationItemSelelectedListener, isto é, a aplicação é desenvolvida baseada num menu do estilo hamburger button, em que a nave- gação realizada na aplicação é efetuada por este menu, em que irão surgir diversos itens de seleção. A aplicação requer o acesso à Internet e verifica logo no início da aplicação se o utilizador tem ligação estabelecida ou não. Caso esteja desligada é pedido ao utilizador que ative a conexão.

6.1.2 GPS e Maps

Na classe GPS realiza-se a obtenção da localização. Na classe Maps é feito a obtenção das coordenadas, latitude e longitude, através da localização obtida na classe GPS.

6.1.3 Photo

Esta classe permite ao Cliente fotografar e tem a possibilidade de pré-visualizar a fotografia que acabou de tirar. De forma a conseguir capturar imagens, a aplicação necessita de ter a localização ativa. Caso não se verifique é pedido ao utilizador que estabelece a conexão.

6.1.4 Send

A classe Send realiza o upload de imagens para a base de dados da aplicação. Após a imagem ser selecionada da galeria é gerado uma pré-visualização, de forma que o utilizador observe a fotografia que selecionou.

6.1.5 RegistarVeiculos

Nesta classe o utilizador pode proceder ao registo dos seus veículos que lhe tenham sido furtados, introduzindo as informações que considere necessárias.

6.1.6 VerMeusVeiculos

A classe VerMeusVeiculos permitir ao que Cliente possa visualizar, editar ou remover os seus veículos existentes. Caso a intenção do utilizador seja remover o seu veículo é lhe questionado mais que uma vez se deseja prosseguir de forma a prevenir alguma ação involuntária em que pudesse resultar na eliminação indesejada do veículo.

6.1.7 RegistarAnuncios

Nesta classe o utilizador pode proceder ao registo dos seus anúncios, referindo a localidade em qual aconteceu o furto, respectiva recompensa monetária que esteja disposto a oferecer a quem recuperar o seu veículo e outras informações que considere pertinentes.

6.1.8 VerMeusAnuncios

A classe VerMeusVeiculos permitir ao que Cliente possa visualizar, editar ou remover os seus anúncios existentes. Caso a intenção do utilizador seja remover o seu anúncio é lhe questionado mais que uma vez se deseja prosseguir de forma a prevenir alguma ação involuntária em que pudesse resultar na eliminação indesejada do anúncio.

6.1.9 ProAnun

Esta classe é responsável pela procura de anúncios que venham a ser efetuados pelo utilizador. Efetuará a sua procura, definindo qual localidade pretende consultar.

6.1.10 RegistarClientes

Esta classe permite o registo de clientes, em que estes têm de proceder ao preenchimento de um breve formulário referente as seus dados pessoais.

6.1.11 Login

Classe responsável pelo início de sessão do utilizador.

Capítulo 7

Resultados

Figura 6.1: Distância em metros

Figura 6.2: Distância em centímetros

Figura 6.3: Ocultação da matrícula

```
Request ID: 1

        Send Date: 2017/09/17 Time: 13:14:29
| Localização: Alexy Doce, Rua José António Cruz, Vilar, São Vítor, Braga, Cávado, Norte, Portugal
| Matricula: 9819VJ
| Confiança: 89.923615 %
| Tempo Processamento: 62.654575 ms
| Marca: Seat
| Modelo: Ibiza
| Ano: 2003
| Cor : Cinzento
| Combustível: Diesel
| cv :75
| cc: 1422
        Execution Date: 2017/09/17 Time: 13:14:45
```

Figura 6.4: Resultado da extração

Capítulo 8

Reflexão e Conclusão

O servidor tem a capacidade de extrair matrículas das fotografias de forma constante, e sincronizada, mesmo com uma maior carga de trabalho.

Apesar do seu correto funcionamento, o servidor só se torna viável, se a matrícula extraída no processo for a correta da imagem, algo difícil de controlar, e que está dependente da cor do carro, da luminosidade da fotografia, resolução, estado da matrícula entre outras variáveis.

Como trabalho futuro, existe o desejo de criar um detetor automático de matriculas, usando o OpenCV.

Bibliografia

[1] *Pokémon go*, http://www.pokemongo.com/, Fevereiro de 2017.

[2] *Figura 1.1*, https://play.google.com/store/apps/details?id=com.itzkow.licenceplatereader, Fevereiro de 2017.

[3] *Github*, https://github.com/openalpr/openalpr, Fevereiro de 2017.

[4] *Sapo venda já*, http://auto.sapo.pt/VendaJa, Fevereiro de 2017.

Apêndice A

Proposta Original do Projeto

Curso de Licenciatura em Engenharia Informática
Projeto 3º Ano - Ano letivo de 2016/2017

Análise de Imagens na Detecção de Automóveis Roubados

Orientador: Leonel Domingues Deusdado

1 Objetivo

Desenvolvimento de uma aplicação para smartphone, que permite recuperação de carros roubados, utilizando a tecnologia de Análise de Imagens e Geolocalização.

2 Detalhes

Pretende-se pensar/implementar numa aplicação para smartphone, fazendo o uso da câmara para que consiga numa 1ª fase detectar e reconhecer a matrícula do veículo roubado.
A aplicação permite que a um utilizador cujo veículo foi roubado, se registe na aplicação preenchendo um formulário com a descrição do veículo, nomeadamente, marca, modelo, cor, matrícula e alguns detalhes pertinentes para a identificação do veículo, assim como a última localização do mesmo. O utilizador pode atribuir de forma opcional uma recompensa monetária pela respetiva localização do seu veículo.
O utilizador na sua conta pode publicar anúncios de carros roubados ou pode procurar anúncios de outros utilizadores, sendo que a matrícula dos carros não se encontra visível nos anúncios.
Se um dado utilizador pretender proceder à procurar de um determinado veículo, apenas tem de fotografar a matrícula desse veículo.
Numa 2ª fase, essa matrícula é "retirada" da imagem fotografada, e comparada com as matrículas de carros roubados presentes na base de dados.
Se a matrícula coincidir com uma das matriculas existentes na base de dados, automaticamente é enviada uma mensagem para o dono do veículo com a respetiva localização.
Numa fase final, depois de enviada a localização do veículo, é transferida de forma automática a recompensa oferecida no anúncio ao utilizador que encontrou o mesmo.
O projeto irá basear-se em técnicas de detecção de imagens através das imagens reais provenientes da câmara e uso do GPS.

3 Metodologia de trabalho

- Análise de requisitos do problema.
- Estudo da arte da tecnologia a implementar.
- Prototipagem e modelo de desenvolvimento.
- Implementação, testes de interface e ensaios globais ao sistema e à sua caracterização.
- Escrita do relatório.

Apêndice B

Outro(s) Apêndice(s)

Listing B.1: Main class

```
1  public class main {
2     // Webservice host and port
3     private final static int port = 9998;
4     private final static String host = 'http://localhost/';
5
6     public static void main(String[] args) {
7
8        // Inicia os servicos de rede TOR e Privoxy
9        NetworkServices.getCredencials();
10       NetworkServices.startTorService(); // Connect to tor service
11       NetworkServices.startPrivoxyService(); // start privoxy service
12       NetworkServices.setProxyServer(); // Set tor default proxy.
13       if (NetworkServices.statusTorService() == true && ↔
            NetworkServices.statusPrivoxyService() == true) {
14          System.out.println('Network services started with success!');
15       } else {
16          System.out.println('Nertwork services not started!');
17       }
18
19       try {
20          // Inicia a Base de dados
21          DataBase db = new DataBase();
22          db.startDerbyNet();
23
24          System.out.println('DerbyDB started with success!');
25       } catch (Exception e) {
26          System.out.println('DerbyDB  not started!');
27       }
28
29       ListOfRequests listofrequests = new ListOfRequests();
30       listofrequests.multiThreading();
31       // Cria um HTTP SERVER associado a um webservice rest.
32       URI baseUri = UriBuilder.fromUri(host).port(port).build();
33       ResourceConfig config = new ResourceConfig(AlprServerRemote.↔
            class);
34       // Registering an object that was instancied from ↔
            AlprServerRemote class
```

```
35        config.register(new AbstractBinder() {
36
37           @Override
38           protected void configure() {
39              // TODO Auto-generated method stub
40              bind(listofrequests).to(ListOfRequests.class);
41           }
42        });
43        HttpServer server = JdkHttpServerFactory.createHttpServer(↩
             baseUri, config);
44        HttpContext context = server.createContext('/Authentication', ↩
             new AlprServerRemote());              context.
45        setAuthenticator(new BasicAuthenticator('         Authentication'↩
             ) {
46
47           @Override
48           public boolean checkCredentials(String username, String ↩
                password) {
49              // TODO Auto-generated method stub
50              if (username.equals('auth-user') && passwor    d.equals('auth-↩
                   pass')) {
51                 return true;
52              }
53              return false;
54           }
55        });
56
57        // Testa a coneccao do webservice.
58        try {
59           URL url = new URL('http://localhost:9998/WebService/↩
                testConnection');
60           HttpURLConnection connection = (HttpURLConnection) url.↩
                openConnection();
61           connection.getResponseCode();
62
63           System.out.println('HTTPServer started with success!');
64
65        } catch (Exception e) {
66           System.out.println('HTTPServer not started!');
67        }
68
69        listofrequests.take();
70     }
71  }
```

Listing B.2: ListOfImages class

```
1  public class ListOfRequests {
2
3
4     private Request rq;
5     private LinkedBlockingQueue<Request> requestList;
6     private LinkedBlockingQueue<Request> executedRequestList;
7
8     //Instancia as filas de pedidos pendentes e de pedidos executados
9     public ListOfRequests()
10    {
11       this.requestList = new LinkedBlockingQueue<>(64);
12       this.executedRequestList = new LinkedBlockingQueue<>(64);
```

```
13      }
14
15      //Cria threads consuante os cores do CPU para serem utilizados nas ↔
            filas instanciadas.
16      public void multiThreading()
17      {
18          Thread[] threads = new Thread[Runtime.getRuntime().↔
                availableProcessors()];
19          for (int i = 0; i < threads.length; i++) {
20              threads[i] = new Thread(new ListOfImages(this.requestList, ↔
                    this.executedRequestList));
21              threads[i].start();
22          }
23      }
24
25      // Insere um novo pedido na fila.
26      public void offer(Request r)
27      {
28          this.requestList.offer(r);
29      }
30
31      //Retira um pedido da fila de pedidos executados.
32      public void take()
33      {
34          while(true)
35          {
36              try {
37
38                  this.rq = this.executedRequestList.take();
39                  System.out.println('\n Request ID: '+rq.getUserId());
40                  System.out.println('\n\t Send Date: '+rq.getSendDate()+' ↔
                        Time: '+rq.getSendTime());
41
42                  //Insere na tabela envios da base de dados e guarda o id ↔
                        desse registo.
43                  int key = 0;
44                  if(rq.getRequestType() == 1)
45                  {
46                      key = insertSend(rq.getSendDate(), rq.getSendTime(), ↔
                            rq.getExecutionDate(), rq.getExecutionTime(), rq.↔
                            getUserId());
47                  }
48                  for(Result rs:this.rq.getResultList()) //Percorre todas os↔
                        resultados do pedido executado.
49                  {
50
51                     //Insere os resultados na tabela resultados e as ↔
                           respetivas fotos.
52                     if(rq.getRequestType() == 1)
53                     {
54                        int key2 = insertPhotographyData(rs.getDate(), rs.↔
                              getTime(), rs.getGeolocation(), rs.getLatitude(),↔
                              rs.getLongitude(), rs.getPlate(), rs.↔
                              getConfidencePlate(), rs.getBrand(), rs.getModel↔
                              (), Integer.parseInt(rs.getYear()), rs.getColor()↔
                              , rs.getFuel(), Integer.parseInt(rs.getHP()), ↔
                              Integer.parseInt(rs.getCC()), key);
55                                 insertPhotography(key2 ,rs.getImagePath());
56                                 Search se = new Search(key2, rs.getPlate())↔
                                       ;
```

```
57                  }
58                  else
59                  {
60                      //Caso o pedido seja do tipo 0 insere o novo veiculo↔
                            .
61                      insertVehicle(rs.getImagePath(), rs.getPlate(), rs.↔
                            getBrand(), rs.getModel(), Integer.parseInt(rs.↔
                            getYear()), rs.getColor(), rs.getFuel(), Integer.↔
                            parseInt(rs.getHP()), Integer.parseInt(rs.getCC()↔
                            ), rq.getUserId());
62
63                  }
64                  //Apaga os ficheiros temporarios criados.
65                  Files.delete(Paths.get(rs.getImagePath()));
66              }
67              System.out.println('\t Execution Date: '+rq.↔
                    getExecutionDate()+' Time: '+rq.getExecutionTime());
68
69          } catch (Exception e) {
70              // TODO Auto-generated catch block
71              //e.printStackTrace();
72          }
73
74      }
75  }
76
77  //Insere um envio na tabela envios da base de dados, retornando o ↔
        id do registo inserido.
78  private int insertSend(String send_date, String send_time, String ↔
        executed_date, String executed_time, int clientid) throws ↔
        SQLException
79  {
80      int key;
81      Send s = new Send();
82          s.setSendDate(send_date);
83          s.setSendTime(send_time);
84          s.setExecutedDate(executed_date);
85          s.setExecutedTime(executed_time);
86          s.setClientId(clientid);
87      tblSend ts = new tblSend();
88      key =(int)ts.insert(s);
89      ts.closeConnection();
90      return key;
91  }
92
93  //Insere um resultado na tabela PhotographyData retornando o id do ↔
        registo inserido.
94  private int insertPhotographyData(String date, String time, String ↔
        geolocation, double latitude, double longitude, String plate, ↔
        float confidence, String brand, String model, int ↔
        registration_year, String color, String fuel, int hp, int cc, ↔
        int sendid) throws SQLException
95  {
96      int key;
97
98          PhotographyData pd = new PhotographyData();
99              pd.setDate(date);
10              pd.setTime(time);
11              pd.setGeolocation(geolocation);
12              pd.setLatitude(latitude);
```

```
103             pd.setLongitude(longitude);
104             pd.setPlate(plate);
105             pd.setConfidence(confidence);
106             pd.setBrand(brand);
107             pd.setModel(model);
108             pd.setRegistrationYear(registration_year);
109             pd.setColor(color);
110             pd.setFuel(fuel);
111             pd.setHP(hp);
112             pd.setCC(cc);
113             pd.setSendId(sendid);
114         tblPhotographyData tpd = new tblPhotographyData();
115             key = (int)tpd.insert(pd);
116             tpd.closeConnection();
117             return key;
118     }
119
120     //Insere uma fotografia, associada ao resultado.
121     private void insertPhotography(int photographydataid, String ↔
            imagepath) throws IOException, SQLException //inserts blob ↔
            pictures into database.
122     {
123         Photography p = new Photography();
124             p.setId(photographydataid);
125             p.setPhoto(imagepath);
126         tblPhotography tp = new tblPhotography();
127             tp.insert(p);
128             tp.closeConnection();
129     }
130
131     //Insere um novo veiculo, tendo em conta se ele ja existe ou nao.
132     private void insertVehicle(String imagepath, String plate, String ↔
            brand, String model, int registration_year, String color, String↔
            fuel, int hp, int cc, int clientid) throws SQLException, ↔
            IOException
133     {
134         Vehicle v = new Vehicle();
135             v.setPlate(plate);
136             v.setBrand(brand);
137             v.setModel(model);
138             v.setRegistrationYear(registration_year);
139             v.setColor(color);
140             v.setFuel(fuel);
141             v.setHP(hp);
142             v.setCC(cc);
143             v.setClientId(clientid);
144         tblVehicle tv = new tblVehicle();
145             ResultSet rs = tv.checkUnique(plate);
146                 rs.next();
147                 if( rs.getInt(1) == 0)
148                 {
149                     //Caso nao exista insere o novo veiculo e a foto ↔
                        correspondente.
150                     insertPhotographyVehicle(imagepath, (int) tv.insert(↔
                        v));
151                 }
152                 else
153                 {
154                     // Caso ja exista verifica se ele ta desativado, para↔
                        ativar.
```

```
155                     ResultSet rs2 = tv.selectClient_v2(clientid);
156                     while(rs2.next())
157                     {
158                        if(rs2.getString(2).equalsIgnoreCase(plate))
159                        {
160                           insertPhotographyVehicle(imagepath, rs2.getInt↩
                              (1));
161                           if(rs2.getBoolean(12) == false)
162                           {
163                              tv.activateVehicle(rs2.getInt(1));
164                           }
165                        }
166                     }
167                     rs2.close();
168                  }
169               rs.close();
170            tv.closeConnection();
171      }
172
173      //Insere as fotos referentes a veiculos na tabela ↩
            PhotographyVehicle.
174      private void insertPhotographyVehicle(String imagepath, int ↩
            vehicleid) throws SQLException, IOException
175      {
176            PhotographyVehicle pv = new PhotographyVehicle();
177               pv.setPhoto(imagepath);
178               pv.setVehicleId(vehicleid);
179            tblPhotographyVehicle tpv = new tblPhotographyVehicle();
180               tpv.insert(pv);
181               tpv.closeConnection();
182      }
183
184   }
```

Listing B.3: CarService class

```
1   public class CarService {
2
3      //Connection attributes
4      private String regExpr;
5      private String urlPage1;
6      private String urlPage2;
7      private List<String> cookies;
8      private HttpsURLConnection conn;
9      private final String USER_AGENT = 'Mozilla/5.0';
10
11     //Car attributes
12     private static String plate;
13     private static String brand;
14     private static String model;
15     private static String year;
16     private static String fuel;
17     private static String color;
18     private static String type;
19     private static String cc;
20     private static String hp;
21
22     public CarService() throws Exception
23     {
```

```
24          this.urlPage1 = 'http://auto.sapo.pt/VendaJa/';
25          this.urlPage2 = 'http://auto.sapo.pt/VendaJa/Vender';
26      }
27      public void Connection() throws Exception
28      {
29        CarService http = new CarService();
30
31        CookieHandler.setDefault(new CookieManager());
32
33        // 1. Send a 'GET' request, so that you can extract the form's ↔
             data.
34        String page = http.GetPageContent(this.urlPage1);
35        String postParams = http.getFormParams(page, this.plate);
36
37
38        // 2. Construct above post's content and then send a POST request↔
               for
39        // authentication
40        http.sendPost(this.urlPage1, postParams);
41
42        // 3. success then go to gmail.
43        String result = http.GetPageContent(this.urlPage2);
44        //System.out.println(result);
45      }
46
47
48      private String getFormParams(String html, String plate) throws ↔
           Exception {
49      // TODO Auto-generated method stub
50         System.out.println('Extracting form's data...');
51
52         Document doc = Jsoup.parse(html);
53
54         // Google form id
55         Element loginform = doc.getElementById('IndexFormBig');
56         Elements inputElements = loginform.getElementsByTag('input');
57         List<String> paramList = new ArrayList<String>();
58         for (Element inputElement : inputElements) {
59            String key = inputElement.attr('name');
60            String value = inputElement.attr('value');
61
62            if (key.equals('RegistrationPlate'))
63               value = plate;
64
65            paramList.add(key + '=' + URLEncoder.encode(value, 'UTF-8'));
66         }
67
68         // build parameters list
69         StringBuilder result = new StringBuilder();
70         for (String param : paramList) {
71            if (result.length() == 0) {
72               result.append(param);
73            } else {
74               result.append('&' + param);
75            }
76         }
77
78         return result.toString();
79      }
80
```

```
81
82      private void sendPost(String url, String postParams) throws ↩
            Exception {
83      // TODO Auto-generated method stub
84
85          URL obj = new URL(url);
86          HttpURLConnection conn = (HttpURLConnection) obj.openConnection↩
                ();
87          // Acts like a browser
88          conn.setUseCaches(false);
89          conn.setRequestMethod('POST');
90          conn.setRequestProperty('Host', 'auto.sapo.pt');
91          conn.setRequestProperty('User-Agent', USER_AGENT);
92          conn.setRequestProperty('Accept',
93              'text/html,application/xhtml+xml,application/xml;q=0.9,*/*↩
                    ;q=0.8');
94          conn.setRequestProperty('Accept-Language', 'en-US,en;q=0.5');
95          for (String cookie : this.cookies) {
96              conn.addRequestProperty('Cookie', cookie.split(';', 1)[0])↩
                    ;
97          }
98          conn.setRequestProperty('Connection', 'keep-alive');
99          conn.setRequestProperty('Referer', 'http://auto.sapo.pt/↩
                VendaJa/');
100         conn.setRequestProperty('Content-Type', 'application/x-www-↩
                form-urlencoded');
101         conn.setRequestProperty('Content-Length', Integer.toString(↩
                postParams.length()));
102
103         conn.setDoOutput(true);
104         conn.setDoInput(true);
105
106     // Send post request
107     DataOutputStream wr = new DataOutputStream(conn.getOutputStream↩
            ());
108         wr.writeBytes(postParams);
109         wr.flush();
110         wr.close();
111
112     int responseCode = conn.getResponseCode();
113         System.out.println('\nSending 'POST' request to URL : ' + url↩
                );
114         System.out.println('Post parameters : ' + postParams);
115         System.out.println('Response Code : ' + responseCode);
116
117   //System.out.println('##########################################↩
          CODE##############################');
118         BufferedReader in =
119                   new BufferedReader(new InputStreamReader(conn.↩
                        getInputStream()));
120         String inputLine;
121         StringBuffer response = new StringBuffer();
122         int line = 0;
123         while ((inputLine = in.readLine()) != null) {
124             line++;
125             response.append(inputLine);
126
127             if(line == 250 && inputLine.contains('<h4>Nao encontramos ↩
                    informacao sobre a sua matricula.<br/> Por favor ↩
                    introduza os dados da viatura manualmente.</h4>'))
```

```
128             {
129                 System.out.println('No data found!');
130                 break;
131             }
132             else
133             {
134                 switch (line) {
135                     case 256:
136                           this.cc = regularExpression(inputLine, '<input↔
                                data-val=\'true\' data-val-number=\'The ↔
                                field CC must be a number.\' id=\'CC\' name↔
                                =\'CC\' type=\'hidden\' value=\'(.+?)\'');
137                           this.hp = regularExpression(inputLine,'<input ↔
                                data-val=\'true\' data-val-number=\'The ↔
                                field HP must be a number.\' id=\'HP\' name↔
                                =\'HP\' type=\'hidden\' value=\'(.+?)\' ↔
                                />');
138                     break;
139                     case 261:
140                           this.brand = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray no_space_bottom\'>(.+?)</p↔
                                >');
141                     break;
142                     case 268:
143                           this.model = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray no_space_bottom\'>(.+?)</p↔
                                >');
144                     break;
145                     case 277:
146                           this.year = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray no_space_bottom\'>(.+?)</p↔
                                >');
147                     break;
148                     case 284:
149                           this.fuel = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray no_space_bottom\'>(.+?)</p↔
                                >');
150                     break;
151                     case 292:
152                           this.color = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray no_space_bottom\'>(.+?)</p↔
                                >');
153                     break;
154                     case 301:
155                           this.type = regularExpression(inputLine,'<p ↔
                                class=\'bg_gray\'>(.+?)</p>');
156                     break;
157                 }
158             }
159         }
160         in.close();
161         // System.out.println(response.toString());
162     // System.out.println('##########################################↔
        End CODE ############################');
163     }
164
165
166     private String GetPageContent(String url) throws Exception {
167     // TODO Auto-generated method stub
168         URL obj = new URL(url);
```

```
169         HttpURLConnection conn = (HttpURLConnection) obj.openConnection↩
               ();
170
171            // default is GET
172            conn.setRequestMethod('GET');
173            conn.setUseCaches(false);
174
175            // act like a browser conn.setRequestProperty('User-
176            Agent', USER_AGENT); conn.setRequestProperty('
177            Accept',
178               'text/html,application/xhtml+xml,application/xml;q=0.9,*/↩
                  *;q=0.8');
179            conn.setRequestProperty('Accept-Language', 'en-US,en;q=0.5');
180            if (cookies != null) {
181               for (String cookie : this.cookies) {
182                  conn.addRequestProperty('Cookie', cookie.split(';', 1)↩
                     [0]);
183               }
184            }
185
186         int responseCode = conn.getResponseCode();
187            System.out.println('\nSending 'GET' request to URL : ' + url)↩
               ;
188            System.out.println('Response Code : ' + responseCode);
189
190      // System.out.println('################################################↩
         CODE ##############################');
191            BufferedReader in =
192                     new BufferedReader(new InputStreamReader(conn.↩
                        getInputStream()));
193            String inputLine;
194            StringBuffer response = new StringBuffer();
195
196            while ((inputLine = in.readLine()) != null) {
197               response.append(inputLine);
198            }
199            in.close();
200      // System.out.println('################################################↩
         End CODE ##############################');
201
202            // Get the response cookies
203            setCookies(conn.getHeaderFields().get('Set-Cookie'));
204
205            return response.toString();
206      }
207      public List<String> getCookies() {
208         return cookies;
209      }
210      public void setCookies(List<String> cookies) {
211         this.cookies = cookies;
212      }
213
214      private String regularExpression(String inputLine, String regExpr)
215      {
216            final Pattern pattern = Pattern.compile(regExpr);
217            final Matcher matcher = pattern.matcher(inputLine);
218            matcher.find();
219            //System.out.println(matcher.group(1));
220            return matcher.group(1);
221      }
```

```
222
223     //set
224     public void setPlate(String plate)
225     {
226         this.plate = plate;
227     }
228     public String getPlate()
229     {
230         return this.plate;
231     }
232     public String getBrand()
233     {
234         return this.brand;
235     }
236     public String getModel()
237     {
238         return this.model;
239     }
240     public String getYear()
241     {
242         return this.year;
243     }
244     public String getFuel()
245     {
246         return this.fuel;
247     }
248     public String getColor()
249     {
250         return this.color;
251     }
252     public String getType()
253     {
254         return this.type;
255     }
256     public String getCC()
257     {
258         return this.cc;
259     }
260     public String getHP()
261     {
262         return this.hp;
263     }
264 }
```

Listing B.4: Geolocation class

```
1  public class GeolocationService {
2
3      private Double latitude;
4      private Double longitude;
5
6      private String geolocation;
7      private String countrycode;
8      private String country;
9      private String postcode;
10     private String state;
11     private String county;
12     private String city;
13     private String suburb;
```

```
14      private String road;
15
16
17
18      public GeolocationService(Double latitude, Double longitude)
19      {
20          this.latitude = latitude;
21          this.longitude = longitude;
22          getLocation();
23      }
24      private void getLocation()
25      {
26
27          NominatimReverseGeocodingJAPI nominatim1 = new ↔           N
                ominatimReverseGeocodingJAPI(); //create instance with ↔
                default zoom level (18)
28              setGeolocation(nominatim1.getAdress(this.latitude, this.↔
                    longitude).getDisplayName());//returns Address object for ↔
                    the given position
29              setCountryCode(nominatim1.getAdress(this.latitude, this.↔
                    longitude).getCountryCode());
30              setCountry(nominatim1.getAdress(this.latitude, this.longitude↔
                    ).getCountry());
31              setPostCode(nominatim1.getAdress(this.latitude, this.↔
                    longitude).getPostcode());
32              setState(nominatim1.getAdress(this.latitude, this.longitude).↔
                    getState());
33              setCounty(nominatim1.getAdress(this.latitude, this.longitude)↔
                    .getCounty());
34              setCity(nominatim1.getAdress(this.latitude, this.longitude).↔
                    getCity());
35              setSuburb(nominatim1.getAdress(this.latitude, this.longitude)↔
                    .getSuburb());
36              setSuburb(nominatim1.getAdress(this.latitude, this.longitude)↔
                    .getRoad());
37
38      }
39      //set
40      private void setGeolocation(String geolocation)
41      {
42          this.geolocation = geolocation;
43      }
44      private void setCountryCode(String countrycode)
45      {
46          this.countrycode = countrycode;
47      }
48      private void setCountry(String country)
49      {
50          this.country = country;
51      }
52      private void setPostCode(String postcode)
53      {
54          this.postcode = postcode;
55      }
56      private void setState(String state)
57      {
58          this.state = state;
59      }
60      private void setCounty(String county)
61      {
```

```
62          this.county = county;
63      }
64      private void setCity(String city)
65      {
66          this.city = city;
67      }
68
69      private void setSuburb(String suburb)
70      {
71          this.suburb = suburb;
72      }
73      private void setRoad(String road)
74      {
75          this.road = road;
76      }
77
78      //get
79      public String getGeolocation()
80      {
81          return this.geolocation;
82      }
83      public String getCountryCode()
84      {
85          return this.countrycode;
86      }
87      public String getCountry()
88      {
89          return this.country;
90      }
91      public String getPostCode()
92      {
93          return this.postcode;
94      }
95      public String getState()
96      {
97          return this.state;
98      }
99      public String getCounty()
100     {
101         return this.county;
102     }
103     public String getCity()
104     {
105         return this.city;
106     }
107     public String getSuburb()
108     {
109         return this.suburb;
110     }
111     public String getRoad()
112     {
113         return this.road;
114     }
115 }
```

Listing B.5: NetworkServices class

```
1  public class NetworkServices {
2
```

```
3     //Private String that stores root password.
4     private static String password;
5
6     public NetworkServices()
7     {
8
9     }
10
11    public static void getCredencials()
12    {
13       //Code to scan file where the password was stored.
14       try {
15          Scanner scanner = new Scanner( new File('/home/Diogo/↩
               Documentos/Projeto EI/rootpw.txt') );
16          password = scanner.useDelimiter(';').next();
17          scanner.close(); // Put this call in a finally block
18       } catch (FileNotFoundException e) {
19          // TODO Auto-generated catch block
20          e.printStackTrace();
21       }
22    }
23
24    //#####################################TOR service↩
       ##############################
25    public static void startTorService()
26    {
27         try {
28          String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
               | sudo -S service tor start'};
29          Process proc = Runtime.getRuntime().exec(command);
30             proc.waitFor();
31         }
32         catch (Exception e) {
33         }
34    }
35    public static void stopTorService()
36    {
37         try {
38          String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
               | sudo -S service tor stop'};
39          Process proc = Runtime.getRuntime().exec(command);
40             proc.waitFor();
41         }
42         catch (Exception e) {
43         }
44    }
45    public static void restartTorService()
46    {
47         try {
48          String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
               | sudo -S service tor restart'};
49          Process proc = Runtime.getRuntime().exec(command);
50             proc.waitFor();
51         }
52         catch (Exception e) {
53         }
54    }
55    public static boolean statusTorService()
56    {
57       boolean state = false;
```

```
58        String information = null;
59        try {
60           String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
                | sudo -S service tor status'};
61           Process proc = Runtime.getRuntime().exec(command);
62              proc.waitFor();
63
64             Buffered Reader std Input = new Buffered Reader ( new ↩
                  InputStreamReader(proc.getInputStream()));
65             while ((information = stdInput.readLine()) != null) {
66               //  System.out.println(info);
67              if(information.contains('active (running)'))
68              {
69                 //System.out.println(information);
70                 state = true;
71              }
72             }
73         }
74        catch (Exception e) {
75                 System.out.println('exception happened - here's what I ↩
                     know :');
76                 e.printStackTrace();
77                 System.exit(-1);
78         }
79        return state;
80     }
81
82     //#######################################Privoxy service↩
          ################################
83     public static void startPrivoxyService()
84     {
85          try {
86           String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
                | sudo -S service privoxy start'};
87           Process proc = Runtime.getRuntime().exec(command);
88                 proc.waitFor();
89           }
90          catch (Exception e) {
91          }
92     }
93     public static void stopPrivoxyService()
94     {
95          try {
96           String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
                | sudo -S service privoxy stop'};
97           Process proc = Runtime.getRuntime().exec(command);
98                  proc.waitFor();
99           }
100         catch (Exception e) {
101         }
102    }
103    public static void restartPrivoxyService()
104    {
105         try {
106          String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
                | sudo -S service privoxy restart'};
107          Process proc = Runtime.getRuntime().exec(command);
108                proc.waitFor();
109          }
110         catch (Exception e) {
```

```
111         }
112     }
113     public static boolean statusPrivoxyService()
114     {
115        boolean state = false;
116        String information = null;
117        try {
118           String command[] = {'/bin/bash', '-c', 'echo'+' '+password+' ↩
               | sudo -S service privoxy status'};
119           Process proc = Runtime.getRuntime().exec(command);
120              proc.waitFor();
121
122           BufferedReader stdInput = new BufferedReader(new ↩
                 InputStreamReader(proc.getInputStream()));
123             while ((information = stdInput.readLine()) != null) {
124              //  System.out.println(info);
125              if(information.contains('active (running)'))
126              {
127                 //System.out.println(information);
128                 state = true;
129              }
130             }
131         }
132        catch (Exception e) {
133                 System.out.println('exception happened - here's what I ↩
                     know :');
134                 e.printStackTrace();
135                 System.exit(-1);
136         }
137        return state;
138     }
139
140     public static void setProxyServer()
141     {
142              // Code that connects to host http and https proxy.
143         System.setProperty('http.proxyHost', '127.0.0.1');
144          System.setProperty('http.proxyPort', '8118');
145     }
146 }
```

Listing B.6: Image class

```
1  public class Image{
2
3     private static final Alpr alpr = new Alpr('eu', '/home/Diogo/↩
          Documentos/Projeto EI/openalpr/src/build/config/openalpr.conf', ↩
          '/home/Diogo/Documentos/Projeto EI/openalpr/runtime_data');
4     private String image;
5     private AlprResults results;
6
7     public Image(String image)
8     {
9        this.image = image;
10    }
11    public AlprResults alpr()
12    {
13       this.alpr.setTopN(1);
14       synchronized (this.alpr) {
15          try {
```

```
16              this.results =  this.alpr.recognize(image);
17           } catch (AlprException e) {
18              // TODO Auto-generated catch block
19              e.printStackTrace();
20           }
21        }
22        return this.results;
23
24     }
25  }
```

Listing B.7: RegistrationPlateDimensions class

```
1   public class RegistrationPlateCover {
2
3      private int x;
4      private int y;
5      private int width;
6      private int height;
7      private String image;
8
9      public RegistrationPlateCover(String image, ↔
          RegistrationPlateDimensions registrationPlateDimensions)
10     {
11        this.image = image;
12        this.x = registrationPlateDimensions.getX();
13        this.y = registrationPlateDimensions.getY();
14        this.width = registrationPlateDimensions.getWidth();
15        this.height = registrationPlateDimensions.getHeight();
16        draw();
17
18     }
19     private void draw()
20     {
21        BufferedImage imagewrite, imageread;
22        try {
23           imageread = ImageIO.read(new File(image));
24           imagewrite = new BufferedImage(imageread.getWidth(), ↔
                imageread.getHeight(), BufferedImage.TYPE_INT_RGB);
25           imagewrite = imageread;
26
27            // Create a graphics which can be used to draw into the ↔
                buffered image
28            Graphics2D g2d = imagewrite.createGraphics();
29
30            // fill all the image with white
31            g2d.setColor(Color.WHITE);
32            g2d.fillRect(this.x, this.y, this.width, this.height);
33
34            // Disposes of this graphics context and releases any system↔
                 resources that it is using.
35            g2d.dispose();
36
37            // Save as PNG
38            File file = new File(this.image);
39             ImageIO.write(imagewrite, 'jpg', file);
40
41        } catch (IOException e1) {
42           // TODO Auto-generated catch block
```

```
43              e1.printStackTrace();
44          }
45      }
46  }
```

Listing B.8: RegistrationPlateDistances class

```
1   public class RegistrationPlateDistances {
2
3       private CameraParameters cameraParameters;
4       private RegistrationPlateDimensions registrationPlateDimensions;
5       private MetaData imagedata;
6       private Color color;
7
8
9       public RegistrationPlateDistances(RegistrationPlateDimensions ↔
            registrationPlateDimensions, CameraParameters cameraParameters, ↔
            MetaData imagedata)
10      {
11          this.registrationPlateDimensions = registrationPlateDimensions;
12          this.cameraParameters = cameraParameters;
13          this.imagedata = imagedata;
14
15          System.out.println("focal:"+this.cameraParameters.getFocallength↔
                ()+" imageheight:"+this.imagedata.getHeight()+" imagewidth:"+↔
                this.imagedata.getWidth()+" objectwidth:"+↔
                registrationPlateDimensions.getWidth()+" objectheight:"+↔
                registrationPlateDimensions.getHeight()+" sensorHeight:"+↔
                cameraParameters.getSensorHeight()+" sensorWidth:"+↔    camera
                Parameters.getSensorWidth());
16          draw(convertUnits(getDistance()));
17      }
18
19      private BigDecimal getDistance()
20      {
21          int imageWidth = Integer.valueOf(imagedata.getWidth().replaceAll↔
                ("[^0-9]", ""));
22          int imageHeight = Integer.valueOf(imagedata.getHeight().↔
                replaceAll("[^0-9]", ""));
23
24          //check formula type
25          if(imageWidth > imageHeight)
26          {
27              return (new BigDecimal(String.valueOf(this.cameraParameters.↔
                    getFocallength()))).multiply(new BigDecimal("520")).↔
                    multiply(new BigDecimal(String.valueOf(imageWidth)))).↔
                    divide(new BigDecimal(String.valueOf(↔        registration
                    PlateDimensions.getWidth()))).multiply(new ↔ BigDecimal(
                    String.valueOf(cameraParameters.getSensorWidth↔())))), 2,
                    RoundingMode.HALF_UP);
28          }
29          else
30          {
```

```
31          return (new BigDecimal(String.valueOf(this.cameraParameters.↔
               getFocallength())).multiply(new BigDecimal("110")).↔
               multiply(new BigDecimal(String.valueOf(imageHeight)))).↔
               divide(new BigDecimal(String.valueOf(↔
               registrationPlateDimensions.getHeight())).multiply(new ↔
               BigDecimal(String.valueOf(cameraParameters.getSensorHeight↔
               ()))), 2, RoundingMode.HALF_UP);
32       }
33    }
34
35    private String convertUnits(BigDecimal distance)
36    {
37       if(distance.compareTo(new BigDecimal("1000")) == 0 || distance.↔
            compareTo(new BigDecimal("1000")) == 1) //if distance is ↔
            bigger then 1 meter
38       {
39          this.color = Color.RED;
40          return String.valueOf(distance.multiply(new BigDecimal("0.001↔
               ")).setScale(2, RoundingMode.HALF_UP))+" m";
41       }
42       else
43       {
44          this.color = Color.GREEN;
45          return String.valueOf(distance.multiply(new BigDecimal("0.1")↔
               ).setScale(2, RoundingMode.HALF_UP))+" cm";
46       }
47    }
48
49    private void draw(String distance)
50    {
51       BufferedImage imagewrite, imageread;
52       try {
53          imageread = ImageIO.read(new File(imagedata.getImagePath()));
54          imagewrite = new BufferedImage(imageread.getWidth(), ↔
               imageread.getHeight(), BufferedImage.TYPE_INT_RGB);
55          imagewrite = imageread;
56
57           // Create a graphics which can be used to draw into the ↔
               buffered image
58           Graphics2D g2d = imagewrite.createGraphics();
59
60           // fill all the image with white
61           g2d.setColor(this.color);
62           g2d.setStroke(new BasicStroke(20));
63           g2d.drawRect(registrationPlateDimensions.getX(), ↔
               registrationPlateDimensions.getY(), ↔
               registrationPlateDimensions.getWidth(), ↔
               registrationPlateDimensions.getHeight());
64
65
66           g2d.setFont(new Font ("Courier New", 1, 100));
67           g2d.drawString(distance, registrationPlateDimensions.getX(),↔
               registrationPlateDimensions.getY()+↔
               registrationPlateDimensions.getHeight()+100);
68
69
70           // Disposes of this graphics context and releases any system↔
                resources that it is using.
71           g2d.dispose();
72
```

```
73              // Save as PNG
74              File file = new File(this.imagedata.getImagePath());
75               ImageIO.write(imagewrite, "jpg", file);
76
77          } catch (IOException e1) {
78             // TODO Auto-generated catch block
79             e1.printStackTrace();
80          }
81
82       }
```

Listing B.9: Anuncios class

Aplicação

```
1  public class Anuncios {
2      private int id;
3      private String date;
4      private String title;
5      private String reward;
6      private String localization;
7      private String description;
8      private int Vehicle_Id;
9
10     public Anuncios(String date, String title, String reward, String ↔
           localization, String description,int Vehicle_Id, int id) {
11         this.date = date;
12          this.title = title;
13          this.reward = reward;
14          this.localization = localization;
15          this.description = description;
16          this.Vehicle_Id = Vehicle_Id;
17          this.id = id;
18     }
19
20
21
22     public int getId(){
23         return id;
24     }
25
26     public void setId(int id) {
27         this.id = id;
28     }
29
30     public String getDate() {
31         return date;
32     }
33
34     public void setDate(String date) {
35         this.date = date;
36     }
37
38     public String getTitle() {
39         return title;
40     }
41
42     public void setTitle(String title) {
43         this.title = title;
```

```
44      }
45
46      public String getReward() {
47          return reward;
48      }
49
50      public void setReward(String reward) {
51          this.reward = reward;
52      }
53
54      public String getLocalization() {
55          return localization;
56      }
57
58      public void setLocalization(String localiz
59          this.localization = localization;
60      }
61
62      public String getDescription() {
63          return description;
64      }
65
66      public void setDescription(String descript
67          this.description = description;
68      }
69      public int getVehicleId() {
70          return Vehicle_Id;
71      }
72
73      public void setVehicleId(int Vehicle_Id) {
74          this.Vehicle_Id = Vehicle_Id;
75      }
76
77  }
```

Listing B.10: AnunciosAdap class

```
1   public class AnunciosAdap extends BaseAdapter {
2       private Context context;
3       private  int layout;
4       private ArrayList<Anuncios> anunciosArrayList;
5
6
7       public AnunciosAdap(Context context, int layout, ArrayList<↩
            Anuncios> anunciosArrayList) {
8           this.context = context;
9           this.layout = layout;
10          this.anunciosArrayList = anunciosArrayList;
11      }
12      @Override
13      public int getCount() {
14          return anunciosArrayList.size();
15      }
16
17      @Override
18      public Object getItem(int i) {
19          return anunciosArrayList.get(i);
20      }
21
```

```
22      @Override
23      public long getItemId(int i) {
24          return i;
25      }
26
27      private class ViewHolder{
28          TextView an_date, title, reward, location, description , vhID
29      }
30      @Override
31      public View getView(int position, View view, ViewGroup vie wGroup)↔
            {
32          View row = view;
33          ViewHolder holder = new ViewHolder();
34
35          if(row == null){
36              LayoutInflater inflater = (LayoutInflater) context .↔
                    getSystemService(Context.LAYOUT_INFLATER_SERVIC E
37              row = inflater.inflate(layout, null);
38
39              holder.an_date= (TextView) row.findViewById(R.id.a dapDate)↔
                    ;
40              holder.title= (TextView) row.findViewById(R.id.ada pTitle);
41              holder.reward= (TextView) row.findViewById(R.id.ad apReward↔
                    );
42              holder.location= (TextView) row.findViewById(R.id. ↔
                    adapLocal);
43              holder.description = (TextView) row.findViewById(R.veh .id .↔
                    Adap Desc);
44              holder.vhID = (TextView) row.findViewById(R.id.ada p IdV);
45              row.setTag(holder);
46          }
47          else {
48              holder = (ViewHolder) row.getTag();
49          }
50
51          Anuncios anuncios = anunciosArrayList.get(position);
52
53          holder.an_date.setText(anuncios.getDate());
54          holder.title.setText(anuncios.getTitle());
55          holder.reward.setText(anuncios.getReward());
56          holder.location.setText(anuncios.getLocalization());
57          holder.description.setText(anuncios.getDescription());
58          holder.vhID.setText(anuncios.getVehicleId());
59
60          return row;
61      }
62  }
```

Listing B.11: GPS class

```
1   public class GPS extends Service implements LocationListener {
2       @Nullable
3
4       private final Context
5       context; boolean enableGps
6                        = false;
7       boolean disableGps = false;
8       boolean networkOn = false;
9
        Location location;
```

```
10      protected LocationManager locationManager;
11
12      public GPS(Context context) {
13          this.context = context;
14      }
15
16      public boolean isGpsReadyToUse() {
17          //this.context.getSystemService is required if we want to use ↔
                in separate class
18          locationManager = (LocationManager) this.context.↔
                getSystemService(Context.LOCATION_SERVICE);
19          return locationManager.isProviderEnabled(LocationManager.↔
                GPS_PROVIDER);
20      }
21
22      //Create a GetLocation Method //
23      public  Location getLocation(){
24          isGpsReadyToUse();
25          try{
26
27              locationManager = (LocationManager) context.↔
                    getSystemService(LOCATION_SERVICE);
28              enableGps = locationManager.isProviderEnabled(↔
                    locationManager.GPS_PROVIDER);
29              networkOn =locationManager.isProviderEnabled(↔
                    locationManager.NETWORK_PROVIDER);
30
31              if(ContextCompat.checkSelfPermission(context, Manifest.↔
                    permission.ACCESS_FINE_LOCATION) == PackageManager.↔
                    PERMISSION_GRANTED
32                      || ContextCompat.checkSelfPermission(context, ↔
                            Manifest.permission.ACCESS_COARSE_LOCATION) == ↔
                            PackageManager.PERMISSION_GRANTED ){
33
34                  if(enableGps){
35                      if(location==null){
36                          locationManager.requestLocationUpdates(↔
                                LocationManager.GPS_PROVIDER, 10000,10,this↔
                                );
37                          if(locationManager!=null){
38                              location = locationManager.↔     getLastK
                                    nownLocation(LocationManager.↔GPS_
                                    PROVIDER);
39                          }
40                      }
41                  }
42                  // if lcoation is not found from GPS than it will ↔
                        found from network //
43                  if(location==null){
44                      if(networkOn){
45                          locationManager.requestLocationUpdates(↔
                                LocationManager.NETWORK_PROVIDER, 10000,10,↔
                                this);
46                          if(locationManager!=null){
47                              location = locationManager.↔     getLastK
                                    nownLocation(LocationManager.↔NETWORK_
                                    PROVIDER);
48                          }
49                      }
50                  }
```

```
51              }
52
53          }catch(Exception e){
54              e.printStackTrace();
55          }
56          return  location;
57      }
58  @Override
59      public IBinder onBind(Intent intent) {
60          return null;
61      }
62
63      @Override
64      public void onLocationChanged(Location location)
65
66      }
67
68      @Override
69      public void onStatusChanged(String s, int i, Bundle bundle) {
70
71      }
72
73      @Override
74      public void onProviderEnabled(String s) {
75          Intent i = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS↔
                );
76          startActivity(i);
77
78      }
79
80      @Override
81      public void onProviderDisabled(String s) {
82
83      }
84
85  }
```

Listing B.12: Login class

```
1   public class Login extends Fragment {
2       View rootView;
3       Button btn;
4       EditText getEmail, getPassword;
5
6       SQL sqlDB;
7       SQLiteDatabase sqLiteDatabase;
8
9       @Override
10      public View onCreateView(LayoutInflater inflater, ViewGroup ↔
            container,
11                               Bundle savedInstanceState) {
12          AlprClientRemote remote = new AlprClientRemote();
13
14          rootView = inflater.inflate(R.layout.fragment_login, container↔
                , false);
15
16          //login btn
17          btn = (Button) rootView.findViewById(R.id.loginBTN);
18
```

```
19          //text boxes
20          getEmail = (EditText) rootView.findViewById(R.id.loginMailBox);
21          getPassword = (EditText) rootView.findViewById(R.id.loginPassBox);
22
23          // access/ read to/ the database
24          sqlDB = new SQL(getContext());
25          sqLiteDatabase = sqlDB.getReadableDatabase();
26
27
28          btn.setOnClickListener(new View.OnClickListener() {
29              @Override
30              public void onClick(View view) {
31                  validate();
32              }
33          });
34
35          return rootView;
36
37      }
38
39      private void validate() {
40
41          if(check ()){
42              if (sqlDB.checkUser(getEmail.getText().toString().trim(),
                    getPassword.getText().toString().trim())) {
43
44                  Toast.makeText(getActivity(),"Bem-vindo!",Toast.
                        LENGTH_SHORT).show();
45                  Intent lockIntent = new Intent(getActivity(),
                        MainActivity.class);
46                  startActivity(lockIntent);
47
48               }
49              }
50              else {
51                  //Toast.makeText(getActivity(),"!",Toast.LENGTH_SHORT)
                        .show();
52                  AlertDialog.Builder builder = new AlertDialog.Builder(
                        getActivity());
53                  builder.setTitle("O e-mail ou a password estao
                        incorrectos!");
54                  builder.setMessage("Por favor, tente novamente!");
55                  builder.setPositiveButton("OK", new DialogInterface.
                        OnClickListener() {
56                      public void onClick(DialogInterface dialogInterface, int i) {
57                          //vazio
58                          //ao ser pressionado o 'ok' recua para o formulario de login,
59                          //para ser novamente preenchido
60                      }
61                  });
62
63                  Dialog alertDialog = builder.create();
64                  alertDialog.setCanceledOnTouchOutside(false);
65                  alertDialog.show();
66
67              }
```

```
68          }
69
70
71      public boolean check(){
72          String ge = getEmail.getText().toString().trim();
73          String gp = getPassword.getText().toString().trim
74          boolean aux = true;
75
76          if(ge.isEmpty()){
77              getEmail.setError("Email invalido!");
78              aux = false;
79          }
80
81          if (!Patterns.EMAIL_ADDRESS.matcher(getEmail.getText
                toString()).matches()){
82              getEmail.setError("Email invalido!");
83              aux = false;
84          }
85
86          if(gp.isEmpty()){
87              getPassword.setError("Password invalida!");
88              aux = false;
89          }
90
91          return aux;
92      }
93
94  }
```

Listing B.13: MainActivity class

```
1   public class MainActivity extends AppCompatActivity implements ↔
        NavigationView.OnNavigationItemSelectedListener {
2
3       DrawerLayout drawerLayout;
4       Toolbar toolbar;
5       FrameLayout frameLayout;
6       NavigationView navigationView;
7
8
9       @Override
10      protected void onCreate(Bundle savedInstanceState)
11          { super.onCreate(savedInstanceState);
12          setContentView(R.layout.activity_main);
13
14          //enable automatically wi-fi connection
15          enableWifi();
16          enableMobileData
17          ();
18
19         // WifiManager wifiManager = (WifiManager) getSystem
               Service(↔Context.WIFI_SERVICE);
20          //wifiManager.setWifiEnabled(true);
21
22
23          Toolbar toolbar = (Toolbar) findViewById(R.id.
24          toolbar); setSupportActionBar(toolbar);
25
```

```
26          //frameLayout = (FrameLayout) findViewById(R.id.content_frame)↔
                ;
27          drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout)↔
                ;
28
29          ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
30                  this, drawerLayout, toolbar, R.string.↔
                        navigation_drawer_open, R.string.↔
                        navigation_drawer_close);
31          drawerLayout.setDrawerListener(toggle);
32          toggle.syncState();
33
34
35          DrawerLayout drawer = (DrawerLayout) findViewById(R.id.↔
                drawer_layout);
36          toggle = new ActionBarDrawerToggle(
37                  this, drawer, toolbar, R.string.navigation_drawer_open↔
                        , R.string.navigation_drawer_close);
38          drawer.setDrawerListener(toggle);
39          toggle.syncState();
40
41          NavigationView navigationView = (NavigationView) findViewById(↔
                R.id.nav_view);
42          navigationView.setNavigationItemSelectedListener(this);
43
44      }
45
46      @Override
47      public void onBackPressed() {
48          DrawerLayout drawer = (DrawerLayout) findViewById(R.id.↔
                drawer_layout);
49          if (drawer.isDrawerOpen(GravityCompat.START)) {
50              drawer.closeDrawer(GravityCompat.START);
51          } else {
52              super.onBackPressed();
53          }
54      }
55
56
57      @Override
58      public boolean onCreateOptionsMenu(Menu menu) {
59          // Inflate the menu; this adds items to the action bar if it ↔
                is present.
60          getMenuInflater().inflate(R.menu.main, menu);
61          return true;
62      }
63
64      @Override
65      public boolean onOptionsItemSelected(MenuItem item) {
66          // Handle action bar item clicks here. The action bar will
67          // automatically handle clicks on the Home/Up button, so long
68          // as you specify a parent activity in AndroidManifest.xml.
69          int id = item.getItemId();
70
71          //noinspection SimplifiableIfStatement
72          if (id == R.id.action_settings) {
73              return true;
74          }
75
76          return super.onOptionsItemSelected(item);
```

```
77      }
78
79      private void displaySelectedScreen(int itemId) {
80
81          // creating fragment object
82          Fragment fragment = null;
83          String title="";
84
85          //initializing the fragment object which is selected
86          switch(itemId) {
87
88              //gestao de ficheiros R.id.      afar
89              case nav_camera://fotogr
90                  fragment = new Photo();
91                  title = "Fotografar";
92                  /*if(Session.exist()) {
93
94                  }else{                          ctivity(),"!",Toast.↩
95                        //Toast.makeText(getA();
                            LENGTH_SHORT).showuilder = new AlertDialog.↩
96                  AlertDialog.Builder b
                        Builder(this);     tem sessao iniciada
97                  builder.setTitle("Nao!"); tton("OK", new ↩
98                  builder.setPositiveBuClickListener() {
                        DialogInterface.Onck(DialogInterface ↩
99                       public void onCli{ e, int i)
                             dialogInterfac null;
                            Fragment fg =   O;
100                          fg= new Login
101                      }
102                 });                         builder.create();
103                                             edOnTouchOutside(
104                 Dialog alertDialog =        false);
105                 alertDialog.setCancel
106                 alertDialog.show();
107             }*/
108             break;
109
110         case R.id.nav_search://enviar
111             fragment = new Send();
112             title="Enviar";
113           /*  if(Session.exist()) {
114
115             }                               ctivity(),"!",Toast.↩
116             else{                           O;
117                   //Toast.makeText(getAuilder = new AlertDialog.↩
118                     LENGTH_SHORT).show
                  AlertDialog.Builder b tem sessao iniciada
119                     Builder(this);    !"); tton("OK", new ↩
                  builder.setTitle("Nao!ClickListener() {
120               builder.setPositiveBuck(DialogInterface ↩
121                  DialogInterface.On e, int i)
                       public void onCli{ null;
122                        dialogInterfac O;
                           Fragment fg =
123                        fg= new Login
124                  }                          builder.create();
125
126               });
127
128               Dialog alertDialog =
```

```
129                 alertDialog.setCanceledOnTouchOutside(false);
130                 alertDialog.show();
131             }*/
132             break;
133
134
135         //anuncios
136          case R.id.nav_slideshow://procurar anuncios
137             fragment = new ProAnun();         title
138             ="Procurar Anuncios";
139              break;
140
141          case R.id.nav_gallery://registar anuncios
142             fragment = new RegistarAnuncios();
143             title="Registar Anuncios";
144             break;
145
146          case R.id.nav_verAnun://ver os meus anuncios
147             fragment = new VerMeusAnuncios();
148             title="Ver Meus Anuncios";
149             break;
150
151
152         //veiculos
153          case R.id.nav_regVei://registar veiculos
154             fragment = new RegistarVeiculos();
155             title="Registar Veiculos";
156             break;
157
158          case R.id.nav_verVei://ver os meus veiculos
159             fragment = new VerMeusVeiculos();
160             title="Ver Meus Anuncios";
161             break;
162
163
164         //conta
165          case R.id.registar_clientes://registar cliente
166             fragment = new RegistarClientes();
167             title="Registar Clientes";
168             break;
169
170          case R.id.login://registar cliente
171             fragment = new Login();
172             title="Login";
173             break;
174
175         case R.id.nav_send://exit
176             Intent startMain = new Intent(Intent.ACTION_MAIN);
177             startMain.addCategory(Intent.CATEGORY_HOME);
178             startMain.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
179             startActivity(startMain);
180             break;
181
182
183     }
184
185     if (fragment != null) {
186         FragmentTransaction ft = getSupportFragmentManager().↔
              beginTransaction();
187         ft.replace(R.id.content_frame, fragment);
```

```
188             getSupportActionBar().setTitle(title);
189             ft.commit();
190         }
191 
192         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.↩
                drawer_layout);
193         drawer.closeDrawer(GravityCompat.START);
194     }
195 
196     @SuppressWarnings("StatementWithEmptyBody")
197     @Override
198     public boolean onNavigationItemSelected(MenuItem item) {
199         displaySelectedScreen(item.getItemId());
200         return true;
201     }
202 
203     public void enableWifi(){
204         @SuppressLint("WifiManagerLeak") WifiManager wifiManager = (↩
                WifiManager) getSystemService(Context.WIFI_SERVICE);
205         wifiManager.setWifiEnabled(true);
206     }
207 
208     public void enableMobileData() {
209         enableWifi();
210         @SuppressLint("WifiManagerLeak") WifiManager wifiManager = (↩
                WifiManager) getSystemService(Context.WIFI_SERVICE);
211 
212         if (!wifiManager.isWifiEnabled()) {
213             WifiInfo enable = wifiManager.getConnectionInfo();
214             if(enable.getNetworkId()==-1) {
215                 AlertDialog.Builder builder = new AlertDialog.Buil
                        this);                                                  der(↩
216                 builder.setTitle("Network connection not found");
217                 builder.setMessage("Por favor ative a conexao a ↩
                        internet");
218                 builder.setPositiveButton("OK", new DialogInterfac
                        OnClickListener() {                                     e.↩
219                     public void onClick(DialogInterface ↩
                            dialogInterface, int i) {
220                         // Showing Alert Message
                            when the user ↩
221                         Intent intent = new Intent(Settings.↩
                                ACTION_DATA_ROAMING_SETTINGS);
222                         startActivity(intent);
223                     }
224                 });
225 
226                 Dialog alertDialog = builder.create();
227                 alertDialog.setCanceledOnTouchOutside(false);
228                 alertDialog.show();
229             }
230         }
231     }
232 
233 }
```

Listing B.14: Maps class

```
1  public class Maps extends FragmentActivity implements ↔
       OnMapReadyCallback {
2
3      private GoogleMap mMap;
4      private GPS gps;
5      private Location mLocation;
6      double latitude, longitude;
7
8      ExifInterface exif;
9      //private Photo ph;
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_maps);
15         gps = new GPS(getApplicationContext());
16         mLocation = gps.getLocation();
17
18
19         //coordenadas
20         latitude = mLocation.getLatitude();
21         longitude = mLocation.getLongitude();
22
23         // Obtain the SupportMapFragment and get notified when the map↔
               is ready to be used.
24         SupportMapFragment mapFragment = (SupportMapFragment) ↔
               getSupportFragmentManager()
25                 .findFragmentById(R.id.map);
26         mapFragment.getMapAsync(this);
27
28     }
29
30     /**
31      * Manipulates the map once available.
32      * This callback is triggered when the map is ready to be used.
33      * This is where we can add markers or lines, add listeners or ↔
            move the camera. In this case,
34      * we just add a marker near Sydney, Australia.
35      * If Google Play services is not installed on the device, the ↔
            user will be prompted to install
36      * it inside the SupportMapFragment. This method will only be ↔
            triggered once the user has
37      * installed Google Play services and returned to the app.
38      */
39     @Override
40     public void onMapReady(GoogleMap googleMap) {
41         mMap = googleMap;
42
43         // Add a marker in Sydney and move the camera
44         LatLng sydney = new LatLng(latitude, longitude);
45         mMap.addMarker(new MarkerOptions().position(sydney).title("↔
               Aqui"));
46         mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
47     }
48
49
50     public void determineGPS(){
51
52         try {
```

```
53              File directory = new File(Environment.↔
                    getExternalStorageDirectory
54                      + File.separator + "MyAppFolder");
55              exif = new ExifInterface(directory + ".jpeg");
56
57              int num1Lat = (int)Math.floor(latitude);
58              int num2Lat = (int)Math.floor((latitude - num1Lat)     *
59              double num3Lat = (latitude - ((double)num1Lat+((do      60);
                    num2Lat/60))) * 3600000;
60
61              int num1Lon = (int)Math.floor(longitude);
62              int num2Lon = (int)Math.floor((longitude - num1Lon ) * 60);
63              double num3Lon = (longitude - ((double)num1Lon+((d ouble)↔
                    num2Lon/60))) * 3600000;
64
65              exif.setAttribute(ExifInterface.TAG_GPS_LATITUDE, num1Lat+↔
                    "/1,"+num2Lat+"/1,"+num3Lat+"/1000");
66              exif.setAttribute(ExifInterface.TAG_GPS_LONGITUDE, num1Lon ↔
                    +"/1,"+num2Lon+"/1,"+num3Lon+"/1000");
67
68
69              if (latitude > 0) {
70                  exif.setAttribute(ExifInterface.TAG_GPS_LATITU DE_REF,
                        "N");                                          ↔
71              } else {
72                  exif.setAttribute(ExifInterface.TAG_GPS_LATITU DE_REF,
                        "S");
73              }
74
75              if (longitude > 0) {
76                  exif.setAttribute(ExifInterface.TAG_GPS_LONGIT UDE_REF
                        "E");                                          ,↔
77              } else {
78                  exif.setAttribute(ExifInterface.TAG_GPS_LONGIT UDE_REF
                        "W");
79              }
80
81              exif.saveAttributes();
82
83          } catch (IOException e) {
84              Log.e("PictureActivity", e.getLocalizedMessage());
85          }
86
87      }
88
89  }
```

Listing B.15: Photo class

```
1   public class Photo extends Fragment {
2
3       private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = ↔
            1888;
4       Button button;// btn_
5       send; ImageView imageView
6       ; private String path;
7       private Maps getGps;
8       public static DatabaseHelper databaseHelper;
9       GPS gps;
```

```
10      Context context;
11      private CameraParameters cp;
12      private Map<String,CameraParameters> values = new HashMap<String,↔
           CameraParameters>();
13
14
15      @Override
16      public View onCreateView(LayoutInflater inflater, ViewGroup ↔
           containe
              r,
17                               Bundle savedInstanceState) {
18
19          final View rootView = inflater.inflate(R.layout.fragment_photo↔
                ,
20                  container, false);
21          enableGps();
22
23          button = (Button) rootView.findViewById(R.id.button2);
24         // btn_send = (Button) rootView.findViewById(R.id.button4);
25          imageView = (ImageView) rootView.findViewById(R.id.imageView);
26
27
28          databaseHelper = new DatabaseHelper(getContext());
29          databaseHelper.queryData("CREATE TABLE IF NOT EXISTS ↔    image
               Storage (id INTEGER PRIMARY KEY AUTOINCREMENT, image ↔BLOB
               )");
30
31          button.setOnClickListener(new View.OnClickListener() {
32              @SuppressLint("NewApi")
33              @Override
34              public void onClick(View view) {
35
36                  try {
37
38                      CameraManager manager = (CameraManager) context.↔
                           getSystemService(Context.CAMERA_SERVICE);
39                      @SuppressLint({"NewApi", "LocalSuppress"}) ↔
                           CameraCharacteristics characteristics =
                           manager↔
                           .getCameraCharacteristics(
                           manager.↔getCameraIdList()
                           [0]);
40
41                      @SuppressLint({"NewApi", "LocalSuppress"}) SizeF ↔
                           sensorSize = characteristics.get(↔              C
                           ameraCharacteristics.SENSOR_INFO_PHYSICAL_SIZE↔
                           );
42                      @SuppressLint({"NewApi", "LocalSuppress"}) float[]↔
                           focalLength = characteristics.get(↔             C
                           ameraCharacteristics.↔                      LEN
                           S_INFO_AVAILABLE_FOCAL_LENGTHS);
43
44                      cp = new CameraParameters();
45                      cp.setFocallength((float) DatatypeConversions.↔
                           focalLength(focalLength[0]));
46                      cp.setSensorWidth((float) DatatypeConversions.↔
                           sensorWidth(sensorSize.getWidth()));
47                      cp.setSensorHeight((float) DatatypeConversions.↔
                           sensorHeight(sensorSize.getHeight()));
48                      Log.d(String.valueOf(cp.getFocallength())+String.↔
                           valueOf(cp.getSensorHeight())+String.valueOf(cp↔
                           .getSensorWidth()),"texto");
49
```

```
50                      databaseHelper = new DatabaseHelper(getContext());
51                      databaseHelper.insertData(
52                              imageByte(imageView)
53                      );
54
55                  }
56                  catch (Exception e){
57                      e.printStackTrace();
58                  }
59                  Intent intent = new Intent(MediaStore.↔
                        ACTION_IMAGE_CAPTURE);
60                  intent.putExtra(MediaStore.EXTRA_OUTPUT,Uri.fromFile(↔
                        createFile()));
61                  startActivityForResult(intent,
62                          CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
63                  galleryAddPic();
64              }
65          });
66
67
68          return rootView;
69      }
70
71      public static byte[] imageByte(ImageView imageView) {
72          Bitmap bitmap = ((BitmapDrawable)imageView.getDrawable()).↔
                getBitmap();
73          ByteArrayOutputStream stream = new ByteArrayOutputStream();
74          bitmap.compress(Bitmap.CompressFormat.JPEG, 100, stream);
75          byte[] byteArray = stream.toByteArray();
76          return byteArray;
77      }
78
79      @Override
80      public void onActivityResult(int requestCode, int resultCode, ↔
            Intent data){
81          super.onActivityResult(requestCode, resultCode, data);
82          if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
83              if (resultCode == Activity.RESULT_OK) {
84                  Bitmap bmp = (Bitmap) data.getExtras().get("data");
85                  imageView.setImageBitmap(bmp);
86              }
87          }
88      }
89
90      //save full-size photo
91      public File createFile() {
92          Bitmap bitmap;
93          bitmap = BitmapFactory.decodeResource(getResources(), R.id.↔
                imageView);
94
95          // Create an image file name
96          String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").↔
                format(new Date());
97          String imageFileName = "picture" + timeStamp + "_";
98          File image = null;
99
10          if (!Environment.getExternalStorageState().equals(Environment.↔
                MEDIA_MOUNTED)) {
11              Log.d("MyApp", "No SDCARD");
12          } else {
```

```
103             File directory = new File(Environment.↔
                    getExternalStorageDirectory()
104                     + File.separator + "MyAppFolder");
105             directory.mkdirs();
106 
107         try {
108             getGps = new Maps();
109             getGps.determineGPS();
110             imageByte(imageView);
111 
112             image = File.createTempFile(
113                     imageFileName,  /* prefix */
114                     ".jpeg",        /* suffix */
115                     directory     /* directory */
116             );
117 
118             values.put(String.valueOf(image), cp);
119         } catch (IOException e) {
120             e.printStackTrace();
121         }
122     }
123 
124         // Save a file: path for use with ACTION_VIEW intents
125         path = image.getAbsolutePath();
126         return image;
127     }
128 
129 
130     public void enableGps() {
131         // Get Location Manager and check for GPS & Network location ↔
                services
132 
133         LocationManager lm = (LocationManager) getActivity().↔
                getSystemService(context.LOCATION
134         if(lm.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
                SERVICE);
135                 !lm.isProviderEnabled(LocationManager.NETWORK_P ROVIDER
                        ↔
                )) {
136             // Build the alert dialog
137             AlertDialog.Builder builder = new AlertDialog.Build
                    getActivity());                                 er(↔
138             builder.setTitle("Servico de Localizacao desativado
                    ");
139             builder.setMessage("Por favor ative o servico de ↔
                    localiazacao e GPS
140             builder.setPositiveButton("OK", new DialogInterface .↔
                    OnClickListener() {
141                 public void onClick(DialogInterface dialogInter face, ↔
                        int i) {
142                     // Show location settings when the user ↔
                            acknowledges the alert dialog
143                     Intent intent = new Intent(Settings.↔
                            ACTION_LOCATION_SOURCE_SETTINGS);
144                     startActivity(intent);
145                 }
146             });
147             Dialog alertDialog = builder.create();
148             alertDialog.setCanceledOnTouchOutside(false);
149             alertDialog.show();
150         }
151     }
152 
```

```
153     private void galleryAddPic() {
154         Intent mediaScanIntent = new Intent(
                Intent.↔ACTION_MEDIA_SCANNER_SCAN_FILE);
155         File f = new File(path);
156         Uri contentUri = Uri.fromFile(f);
157         mediaScanIntent.setData(contentUri);
158         getActivity().sendBroadcast(mediaScanIntent
            );
159     }
160
161
162
163
164 }
```

Listing B.16: ProAnun class

```
1
2  public class ProAnun extends Fragment {
3      View rootView;
4      Button btn;
5      EditText sbox;
6      ListView listView;
7      ArrayList<String> list;
8      AnunciosAdap adapter = null;
9
10     SQLiteHelper sqLiteHelper;
11
12     @Override
13     public View onCreateView(LayoutInflater inflater, ViewGroup ↔
           container,
14                              Bundle savedInstanceState) {
15         AlprClientRemote remote = new AlprClientRemote();
16
17         sqLiteHelper = new SQLiteHelper(getContext());
18         rootView = inflater.inflate(R.layout.fragment_pro_anun,
19                 container, false);
20
21         // sub btn
22         btn = rootView.findViewById(R.id.searchBTN);
23
24         //text boxes
25         sbox = rootView.findViewById(R.id.searchBox);
26         listView = rootView.findViewById(R.id.searchList);
27
28
29
30         btn.setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View view) {
33                 getData(sbox.getText().toString().trim());
34             }
35         });
36
37         listView.setOnItemClickListener(new AdapterView.↔
               OnItemClickListener() {
38             @Override
39             public void onItemClick(AdapterView<?> adapterView, View ↔
                   view, final int position, long l) {
```

```
40                  Cursor c = sqLiteHelper.query("SELECT " + sqLiteHelper↩
                        .KEY_ID + " FROM " + sqLiteHelper.DB_TABLE);
41                  ArrayList<Integer> arrID = new ArrayList<Integer>();
42                  while (c.moveToNext()) {
43                      arrID.add(c.getInt(0));
44                      adapterView.getItemAtPosition(position);
45                  }
46                  see(getActivity(), arrID.get(position));
47              }
48          });
49
50
51          return rootView;
52      }
53
54      public Cursor getData(String localization) {
55          sqLiteHelper = new SQLiteHelper(getContext());
56          ArrayList<String> theList = new ArrayList<>();
57          Cursor cursor = sqLiteHelper.seeData(localization);
58
59          if (validate()) {
60              Toast.makeText(getActivity(), "A procurar...", Toast.↩
                    LENGTH_SHORT).show();
61
62              if (cursor.getCount() != 0) {
63                  while (cursor.moveToNext()) {
64                      theList.add(cursor.getString(cursor.getColumnIndex↩
                            ("localization")));
65                      ListAdapter listAdapter = new ArrayAdapter<>(↩
                            getActivity(), android.R.layout.↩
                            simple_list_item_1, theList);
66                      listView.setAdapter(listAdapter);
67                      Toast.makeText(getActivity(), "Resultados obtidos!↩
                            ", Toast.LENGTH_SHORT).show();
68                  }
69                  return cursor;
70              } else {
71                  Toast.makeText(getActivity(), "Nao possui qualquer ↩
                        anuncio!", Toast.LENGTH_SHORT).show();
72              }
73          }
74
75          cursor.close();
76          return cursor;
77      }
78
79      public boolean validate() {
80          String sb = sbox.getText().toString().trim();
81          boolean aux = true;
82
83          if (sb.isEmpty()) {                    sbox
84              .setError("Localizacao invalida!"); aux
85              = false;
86          }
87          return aux;
88      }
89
90      public void see(Activity activity, final int position) {
91
92          final Dialog dialog = new Dialog(activity);
```

```
93          dialog.setContentView(R.layout.fragment_see_anun);
94          dialog.setTitle("Informacao");
95
96          final EditText edtDate = (EditText) dialog.findViewById(R.id.↔
               seeDate);
97          final EditText edtTitle = (EditText) dialog.findViewById(R.id.↔
               seeTitle);
98          final EditText edtReward = (EditText) dialog.findViewById(R.id↔
              .seeReward);
99          final EditText edtLocatization = (EditText) dialog.↔
               findViewById(R.id.seeLocalization);
100         final EditText edtDescription = (EditText) dialog.findViewById↔
               (R.id.seeDescription);
101
102
103         // set width for dialog
104         int width = (int) (activity.getResources().getDisplayMetrics()↔
               .widthPixels * 0.95);
105         // set height for  dialog
106         int height = (int) (activity.getResources().getDisplayMetrics↔
               ().heightPixels * 0.8);
107         dialog.getWindow().setLayout(width, height);
108         dialog.show();
109
110         String edate= null, etitle= new String(),  ereward = null, ↔
               elocatization= null , edescription = new String();
111
112         try {   //for holding retrieve data from query and store in ↔
               the form of rows
113          Cursor cursor = sqLiteHelper.query("SELECT * " + " FROM " ↔
                + sqLiteHelper.DB_TABLE + " WHERE " + sqLite
               Helper.↔KEY_ID + " ='" + position);
114
115          if (cursor.getCount() != 0) {
116              while (cursor.moveToNext()) {//Move the cursor to the ↔
                    next row.
117                  edate = cursor.getString(cursor.getColumnIndex("↔
                        date "));
118                  etitle = cursor.getString(cursor.getColumnIndex("↔
                        title "));
119                  ereward = cursor.getString(cursor.getColumnIndex("↔
                        reward"));
120                  elocatization = cursor.getString(cursor.↔
                        getColumnIndex("localization"));
121                  edescription = cursor.getString(cursor.↔
                        getColumnIndex("description"));
122
123             }
124             edtDate.setText(edate);
125             edtTitle.setText(etitle);
126             edtReward.setText(ereward);
127             edtLocatization.setText(elocatization);
128             edtDescription.setText(edescription);
129         }
130         cursor.close();
131         } catch (Exception e) {
132             e.printStackTrace();
133         }
134
135         sqLiteHelper.close();
```

```
136         updateListView();
137     }
138
139     public void updateListView() {
140         // get all data from sqlite
141         try {                       stData = new ArrayList<
142             Array List < String >  li>(); teHelper.getAllData
143             Cursor  cursor  =  sq Li();
144             list. clear ();
145             while (cursor.moveToNext()) {                    sor.
146                 listData.add(cur getString(cursor.getColumnIndex("↔
                       localization")));
147             }
148             ListAdapter adapter = new ArrayAdapter<>(getActivity(), ↔
                    android.R.layout.simple_list_item_1, listData);
149             listView.setAdapter(adapter);
150
151         } catch (Exception e) {
152             e.printStackTrace();
153         }
154
155     }
156 }
```

Listing B.17: RegistarAnuncios class

```
1  public class RegistarAnuncios extends Fragment {
2      View rootView;
3      Button btn;
4      EditText andate, title, reward, location, description, vhID;
5      TextView date;
6
7      private DatePickerDialog.OnDateSetListener dateSetListener;
8      private static final String TAG = "RegistarAnuncios";
9
10     SQLiteHelper sqLiteHelper;
11
12
13     @Override
14     public View onCreateView(LayoutInflater inflater, ViewGroup ↔
           container,
15                              Bundle savedInstanceState) {
16
17         rootView  = inflater.inflate(R.layout.fragment_cre_anun,
18                 container, false);
19
20         // sub btn
21         btn = (Button) rootView.findViewById(R.id.btnADD);
22
23         //text boxes
24         andate = (EditText) rootView.findViewById(R.id.dateBox);
25         title = (EditText) rootView.findViewById(R.id.titleBox);
26         reward = (EditText) rootView.findViewById(R.id.rewardBox);
27         location = (EditText) rootView.findViewById(R.id.locationBox);
28         description = (EditText) rootView.findViewById(R.id.↔
               descriptionBox);
29         vhID = (EditText) rootView.findViewById(R.id.vehicleID);
30
31
```

```
32          //datepicker label
33          date = (TextView) rootView.findViewById(R.id.addAnunDate);
34          date.setOnClickListener(new View.OnClickListener() {
35
36              @Override
37              public void onClick(View view) {
38
39                  int year = Calendar.YEAR;
40                  int month = Calendar.MONTH;
41                  int day = Calendar.DAY_OF_MONTH;
42
43                  DatePickerDialog dialog = new DatePickerDialog(
44                          getActivity(), android.R.style.↔
                               Theme_Holo_Light_Dialog_MinWidth,
45                          dateSetListener,
46                          year, month, day);
47                  dialog.getWindow().setBackgroundDrawable(new ↔
                       ColorDrawable(Color.TRANSPARENT));
48                  dialog.show();
49              }
50
51          });
52
53          dateSetListener = new DatePickerDialog.OnDateSetListener() {
54              @Override
55              public void onDateSet(DatePicker datePicker, int year, int↔
                    month, int day) {
56                  month = month + 1;
57                  Log.d(TAG, year + "/" + month + "/" + day);
58
59                  String dateText = year + "/" + month + "/" + day;
60                  andate.setText(dateText);
61              }
62          };
63
64
65
66          dateSetListener = new DatePickerDialog.OnDateSetListener() {
67              @Override
68              public void onDateSet(DatePicker datePicker, int year, int↔
                    month, int day) {
69                  month = month + 1;
70                  Log.d(TAG, year + "/" + month + "/" + day);
71
72                  String dateText = year + "/" + month + "/" + day;
73                  andate.setText(dateText);
74              }
75          };
76
77          btn.setOnClickListener(new View.OnClickListener() {
78              @Override
79              public void onClick(View view) {
80                  addData();
81              }
82          });
83
84          return rootView;
85      }
86
87      public void addData(){
```

```
88          if(validate()) {
89              AlprClientRemote remote = new AlprClientRemote();
90
91              sqLiteHelper = new SQLiteHelper(getContext());
92              boolean insert = sqLiteHelper.insertData(
93                      andate.getText().toString(),
94                      title.getText().toString(),
95                      reward.getText().toString(),
96                      location.getText().toString(),
97                      description.getText().toString(),
98                      vhID.getText().toString()  );
99              if (insert == true) {
100                 Toast.makeText(getActivity(), "Registo do anuncio ↔
                        feito com sucesso!", Toast.LENGTH_SHORT).
                        show();
101                 Intent lockIntent = new Intent(getActivity(), ↔
                        MainActivity.class);
102                 startActivity(lockIntent);
103             }
104         }
105
106         else{
107             Toast.makeText(getActivity(),"Falhou o registo do anuncio!↔
                    ",Toast.LENGTH_SHORT).show();
108         }
109     }
110
111     public boolean validate(){
112         String ad = andate.getText().toString().trim();
113         String tl = title.getText().toString().trim();
114         String r = reward.getText().toString().trim();
115         String l = location.getText().toString().trim();
116         String d = description.getText().toString().trim();
117
118         boolean aux = true;
119
120         if(ad.isEmpty()){
121             andate.setError("Data invalida!");
122             aux = false;
123         }
124
125         if(tl.isEmpty()||tl.length()>12){
126             title.setError("Titulo invalido!");
127             aux = false;
128         }
129
130
131         if(r.isEmpty()){
132             reward.setError("Recompensa invalida!");
133             aux = false;
134         }
135
136         if(l.isEmpty()){
137             location.setError("Localizacao invalida!");
138             aux = false;
139         }
140         if(d.isEmpty()){
141             description.setError("Descricao invalida!");
142             aux = false;
143         }
144
```

```
145
146         return aux;
147     }
148
149 }
```

Listing B.18: RegistarClientes class

```
1  public class RegistarClientes extends Fragment {
2      View rootView;
3      SQL sql;
4
5      Button btn;
6      EditText name, birthDate, contact, email, password, repassword, ↔
           accountBalance;
7
8      TextView date;
9
10     private DatePickerDialog.OnDateSetListener dateSetListener;
11     private static final String TAG = "RegistarClientes";
12
13     @Override
14     public View onCreateView(LayoutInflater inflater, ViewGroup ↔
           container,
15                              Bundle savedInstanceState) {
16
17         rootView = inflater.inflate(R.layout.fragment_
               reg_cli,↔container, false);
18
19         //register btn
20         btn = (Button) rootView.findViewById(R.id.btnAddCli);
21
22         //text boxes
23         name = (EditText) rootView.findViewById(R.id.nameBox);
24         birthDate = (EditText) rootView.findViewById(R.id.birthDateBox↔
               );
25         contact = (EditText) rootView.findViewById(R.id.contactBox);
26         email = (EditText) rootView.findViewById(R.id.emailBox);
27         password = (EditText) rootView.findViewById(R.id.passBox);
28         repassword = (EditText) rootView.findViewById(R.id.repassBox);
29         accountBalance = (EditText) rootView.findViewById(R.id.↔
               moneyBox);
30
31         //datepicker label
32         date = (TextView) rootView.findViewById(R.id.addNascDate);
33         date.setOnClickListener(new View.OnClickListener() {
34
35             @Override
36             public void onClick(View view) {
37
38                 int year = Calendar.YEAR;
39                 int month = Calendar.MONTH;
40                 int day = Calendar.DAY_OF_MONTH;
41
42                 DatePickerDialog dialog = new DatePickerDialog(
43                         getActivity(), android.R.style.↔
                            Theme_Holo_Light_Dialog_MinWidth,
44                         dateSetListener,
45                         year, month, day);
```

```
46                  dialog.getWindow().setBackgroundDrawable(new ↔
                        ColorDrawable(Color.TRANSPARENT));
47                  dialog.show();
48              }
49
50          });
51
52          dateSetListener = new DatePickerDialog.OnDateSetListener() {
53              @Override
54              public void onDateSet(DatePicker datePicker, int year, i nt↔
                    month, int day) {
55                  month = month + 1;
56                  Log.d(TAG, year + "/" + month + "/" + day);
57
58                  String dateText = year + "/" + month + "/" + day;
59                  birthDate.setText(dateText);
60              }
61          };
62
63          btn.setOnClickListener(new View.OnClickListener() {
64              @Override
65              public void onClick(View view) {
66                 addData();
67              }
68          });
69
70          return rootView;
71      }
72
73      public void addData(){
74          if(validate()) {
75              AlprClientRemote remote = new AlprClientRemote();
76
77              sql = new SQL(getContext());
78              boolean insert = sql.insertData(
79                      name.getText().toString(),
80                      birthDate.getText().toString(),
81                      contact.getText().toString(),
82                      email.getText().toString(),
83                      password.getText().toString(),
84                      repassword.getText().toString(),
85                      accountBalance.getText().toString());
86
87              if (insert == true) {
88                  Toast.makeText(getActivity(), "Registo do anuncio ↔
                        feito com sucesso!", Toast.LENGTH_SHORT).show();
89                  Intent lockIntent = new Intent(getActivity(), ↔
                        MainActivity.class);
90                  startActivity(lockIntent);
91              }
92          }
93
94          else{
95              Toast.makeText(getActivity(),"Falhou o registo do anunci
                    ",Toast.LENGTH_SHORT).show();
96          }
97      }
98      public boolean validate(){
99          String na = name.getText().toString().trim();
100         String bd = birthDate.getText().toString();
```

```
101         String c = contact.getText().toString().trim();
102         String em =  email.getText().toString().trim();
103         String pass =  password.getText().toString().trim();
104         String repass =  repassword.getText().toString().trim();
105         String acb =  accountBalance.getText().toString().trim();
106
107         boolean aux = true;
108
109         if(na.isEmpty()){
110             name.setError("Nome invalido!");
111             aux = false;
112         }
113
114         if(bd.isEmpty()){
115             birthDate.setError("Data invalida!");
116             aux = false;
117         }
118
119
120         if(c.isEmpty()){
121             contact.setError("Contacto invalido!");
122             aux = false;
123         }
124
125         if(em.isEmpty()){
126             email.setError("Email invalido!");
127             aux = false;
128         }
129
130
131         if (!Patterns.EMAIL_ADDRESS.matcher(email.getText().toString()↩
                ).matches()){
132             email.setError("Email invalido!");
133             aux = false;
134         }
135
136         if(pass.isEmpty()){
137             password.setError("Password invalida!");
138             aux = false;
139         }
140         if(repass.isEmpty()){
141             repassword.setError("Password invalida!");
142             aux = false;
143         }
144
145         if(!repass.equals(pass)){
146             repassword.setError("Password nao coindide!");
147             aux = false;
148         }
149
150
151         return aux;
152     }
153
154 }
```

Listing B.19: RegistarVeiculos class

```
1 public class RegistarVeiculos extends Fragment {
```

```
2       View rootView;
3       Button btn;
4       EditText matr, brand, model, regYear, color, fuel, hp, cc, descr, ↔
           cliId;
5
6       TextView dateBtn;
7
8       Database db;
9       private DatePickerDialog.OnDateSetListener dateSetListener;
10      private static final String TAG = "RegistarVeiculos";
11
12
13      @Override
14      public View onCreateView(LayoutInflater inflater, ViewGroup ↔
            containe
               r,
15                               Bundle savedInstanceState) {
16
17
18          rootView = inflater.inflate(R.layout.fragment_reg_vehicles,
19                  container, false);
20
21          // sub btn
22          btn = (Button) rootView.findViewById(R.id.btnAdd);
23
24          //text boxes
25          matr = (EditText) rootView.findViewById(R.id.matrBox);
26          brand = (EditText) rootView.findViewById(R.id.brandBox);
27          model = (EditText) rootView.findViewById(R.id.modelBox);
28          regYear = (EditText) rootView.findViewById(R.id.yearBox);
29          color = (EditText) rootView.findViewById(R.id.colorBox);
30          fuel = (EditText) rootView.findViewById(R.id.fuelBox);
31          hp = (EditText) rootView.findViewById(R.id.hpBox);
32          cc = (EditText) rootView.findViewById(R.id.ccBox);
33          descr = (EditText) rootView.findViewById(R.id.↔
               vehDescriptionBox);
34          cliId  = (EditText) rootView.findViewById(R.id.idCliBox);
35
36
37          //datepicker label
38          dateBtn = (TextView) rootView.findViewById(R.id.btnChooseDate)↔
               ;
39
40
41          dateBtn.setOnClickListener(new View.OnClickListener()
            {
42
43              @Override
44              public void onClick(View view) {
45
46                  int year = Calendar.YEAR;
47                  int month = Calendar.MONTH;
48                  int day = Calendar.DAY_OF_MONTH;
49
50                  DatePickerDialog dialog = new DatePickerDialog(
51                          getActivity(), android.R.style.↔
                               Theme_Holo_Light_Dialog_MinWidth,
52                          dateSetListener,
53                          year, month, day);
54                  dialog.getWindow().setBackgroundDrawable(new ↔
                       ColorDrawable(Color.TRANSPARENT));
55                  dialog.show();
```

```
56              }
57
58          });
59
60          dateSetListener = new DatePickerDialog.OnDateSetListener() {
61              @Override
62              public void onDateSet(DatePicker datePicker, int year, int↔
                    month, int day) {
63                  month = month + 1;
64                  Log.d(TAG, year + "/" + month + "/" + day);
65
66                  String dateText = year + "/" + month + "/" + day;
67                  regYear.setText(dateText);
68              }
69          };
70
71          btn.setOnClickListener(new View.OnClickListener() {
72              @Override
73              public void onClick(View view) {
74                  addData();
75              }
76          });
77
78
79          return rootView;
80      }
81
82      public void addData(){
83
84          if(validate()) {
85              AlprClientRemote remote = new AlprClientRemote();
86              db = new Database(getContext());
87              boolean insert = db.insertData(
88                      matr.getText().toString(),
89                      brand.getText().toString(),
90                      model.getText().toString(),
91                      regYear.getText().toString(),
92                      color.getText().toString(),
93                      fuel.getText().toString(),
94                      hp.getText().toString(),
95                      cc.getText().toString(),
96                      descr.getText().toString(),
97                      cliId.getText().toString());
98
99              if (insert == true) {
100                 Toast.makeText(getActivity(), "Registo do
                    anuncio ↔
101                    feito com sucesso!", Toast.LENGTH_SHORT).
                    show(); Intent lockIntent = new Intent(
102                 getActivity(), ↔
103                    MainActivity.class
                    ); startActivity(lock
}                   Intent);
104             }
105             else{
106                 Toast.makeText(getActivity(),"Falhou o registo do anuncio!↔
                    ",Toast.LENGTH_SHORT).show();
107             }
108     }
109
110     public boolean validate(){
111         String mat = matr.getText().toString().trim();
```

```
112         String br = brand.getText().toString().trim();
113         String mo = model.getText().toString().trim();
114         String ry = regYear.getText().toString().trim();
115         String col = color.getText().toString().trim();
116         String fu = fuel.getText().toString().trim();
117         String h = hp.getText().toString().trim();
118         String c = cc.getText().toString().trim();
119         String desc = descr.getText().toString().trim();
120         String cl = cliId.getText().toString().trim();
121
122
123
124         boolean aux = true;
125
126         if(mat.isEmpty()){
127             matr.setError("Matricula invalida!");
128             aux = false;
129         }
130
131         if(br.isEmpty()){
132             brand.setError("Marca invalida!");
133             aux = false;
134         }
135
136
137         if(mo.isEmpty()){
138             model.setError("Modelo invalido!");
139             aux = false;
140         }
141         if(ry.isEmpty()){
142             regYear.setError("Modelo invalido!");
143             aux = false;
144         }
145
146         if(col.isEmpty()){
147             color.setError("Cor invalida!");
148             aux = false;
149         }
150
151         if(fu.isEmpty()){
152             fuel.setError("Combustivel invalido!");
153             aux = false;
154         }
155
156         if(h.isEmpty()){
157             hp.setError("Horse Power invalido!");
158             aux = false;
159         }
160
161         if(c.isEmpty()){
162             cc.setError("Motor invalido!");
163             aux = false;
164         }
165         if(desc.isEmpty()){
166             descr.setError("Descricao invalida!");
167             aux = false;
168         }
169
170         return aux;
171     }
```

```
172 }
```

Listing B.20: Send class

```
1
2  public class Send extends Fragment {
3
4      public final static int PICK_PHOTO_CODE = 1;
5      Button btn,btsend;
6      ImageView imageview;
7
8      private DatabaseHelper dh;
9
10
11     @Override
12     public View onCreateView(LayoutInflater inflater, ViewGroup ↔
           container,
13                              Bundle savedInstanceState) {
14
15         final View rootView = inflater.inflate(R.layout.fragment_send,
16                 container, false);
17         btn = (Button) rootView.findViewById(R.id.button2);
18         btsend = (Button) rootView.findViewById(R.id.sendBTN);
19         imageview = (ImageView) rootView.findViewById(R.id.↔
               imageViewsend);
20
21         btn.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View view) {
24                 chooseFromStorage();
25             }
26         });
27
28         btsend.setOnClickListener(new View.OnClickListener() {
29             @Override
30             public void onClick(View view) {
31
32        AlprClientRemote remote = new AlprClientRemote();
33
34                     dh = new DatabaseHelper(getContext());
35                     try {
36                         dh.getData(imageByte(imageview));
37                         Toast.makeText(getActivity(), "Foto enviada ↔
                               com sucesso com sucesso!", Toast.↔
                               LENGTH_SHORT).show();
38                         Intent lockIntent = new Intent(getActivity(), ↔
                               MainActivity.class);
39                         startActivity(lockIntent);
40
41                     } catch (Exception e) {
42                         e.printStackTrace();
43                         Toast.makeText(getActivity(), "Falhou no envio↔
                               !", Toast.LENGTH_SHORT).show();
44                     }
45
46             }
47         });
48         return rootView;
49     }
```

```
50
51      private void chooseFromStorage() {
52          Intent intent = new Intent(Intent.ACTION_PICK, Uri.parse(↔
               Environment.getExternalStorageDirectory().getPath()));
53          intent.setType("image/*");
54          startActivityForResult(intent,PICK_PHOTO_CODE);
55      }
56
57      @Override
58      public void onActivityResult(int requestCode, int resultCode, ↔
           Intent data){
59          super.onActivityResult(requestCode, resultCode, data);
60          if (requestCode == PICK_PHOTO_CODE) {
61              if (resultCode == Activity.RESULT_OK) {
62                  try {
63                      final Uri imageUri = data.getData();
64                      Bitmap bm =  MediaStore.Images.Media.getBitmap(↔
                           getActivity().getContentResolver(),imageUri);
65                      imageview.setImageBitmap(bm);
66                  }
67                  catch (FileNotFoundException ex) {
68                      ex.printStackTrace();
69                  }
70                  catch (IOException e) {
71                      e.printStackTrace();
72                  }
73              }
74              else {
75                  Toast.makeText(getActivity(), "Nenhuma fotografia ↔
                       selecionada!",Toast.LENGTH_LONG).show();
76              }
77          }
78      }
79
80      public static byte[] imageByte(ImageView imageView) {
81          Bitmap bitmap = ((BitmapDrawable)imageView.getDrawable()).↔
               getBitmap();
82          ByteArrayOutputStream stream = new ByteArrayOutputStream();
83          bitmap.compress(Bitmap.CompressFormat.JPEG, 100, stream);
84          byte[] byteArray = stream.toByteArray();
85          return byteArray;
86      }
87
88  }
```

Listing B.21: Session class

```
1  public class Session {
2          private static String path = "session.txt";
3          private static Context context;
4
5          public Session(Context context)
6          {
7              this.context = context;
8
9          }
10
11         public boolean create(String email, String password){
12
```

```
13              try {
14
15                  OutputStreamWriter outputStreamWriter = new ↔
                        OutputStreamWriter(this.context.openFileOutput
                        (path↔
                        , Context.MODE_PRIVATE));
16                  outputStreamWriter.write(email.concat("\n"));
17                  outputStreamWriter.write(password);
18                  outputStreamWriter.close();
19
20                  return true;
21              } catch (Exception e) {
22                  e.printStackTrace();
23              }
24
25              return false;
26          }
27          public static boolean destroy()
28          {
29
30              try {
31                  context.getFileStreamPath(path).delete();
32                  return true;
33              }
34              catch(Exception e)
35              {
36              }
37              return false;
38          }
39          public static String[] getSession()
40          {
41
42              try {
43
44                  String[] credencials = new String[2];
45                  InputStream inputStream = context.openFileInput(path);
46                  if(inputStream != null)
47                  {
48                      InputStreamReader inputStreamReader = new ↔
                            InputStreamReader(inputStream);
49                      BufferedReader bufferedReader = new BufferedReader↔
                            (inputStreamReader);
50
51                      int i = 0;
52                      String buffer = "";
53                      StringBuilder stringBuilder = new StringBuilder();
54                      while((buffer = bufferedReader.readLine()) != null↔
                            )
55                      {
56                          credencials[i] = buffer;
57                          i++;
58                      }
59                      inputStream.close();
60                      return credencials;
61                  }
62              } catch (Exception e) {
63
64              }
65              return null;
66          }
67          public static boolean exist()
```

```
68          {
69
70              if(context.getFileStreamPath(path).exists())
71              {
72                  return true;
73              }
74              return false;
75          }
76  }
```

Listing B.22: Veiculos class

```
1   public class Veiculos {
2       private int id;
3       private String plate;
4       private String brand;
5       private String model;
6       private int registration_year;
7       private String color;
8       private String fuel;
9       private int hp;
10      private int cc;
11      private String description;
12      private int Client_Id;
13
14      public Veiculos(String plate, String brand, String model, int ↔
            registration_year, String color, String fuel, int hp, int cc, ↔
            String description, int Client_Id, int id){
15          this.plate = plate;
16          this.brand = brand;
17          this.model = model;
18          this.registration_year = registration_year;
19          this.color = color;
20          this.fuel = fuel;
21          this.hp = hp;
22          this.cc = cc;
23          this.color = color;
24          this.description = description;
25          this.Client_Id = Client_Id;
26          this.id = id;
27      }
28
29
30      public int getId() {
31          return id;
32      }
33
34      public void setId(int id) {
35          this.id = id;
36      }
37
38      public String getPlate() {
39          return plate;
40      }
41
42      public void setPlate(String plate) {
43          this.plate = plate;
44      }
45
```

```
46      public String getBrand() {
47          return brand;
48      }
49
50      public void setBrand(String brand) {
51          this.brand = brand;
52      }
53
54      public String getModel() {
55          return model;
56      }
57
58      public void setModel(String model) {
59          this.model = model;
60      }
61
62      public int getRegistrationYear() {
63          return registration_year;
64      }
65
66      public void setRegistrationYear(int registration_year) {
67          this.registration_year = registration_year;
68      }
69
70      public String getColor() {
71          return color;
72      }
73
74      public void setColor(String color) {
75          this.color = color;
76      }
77
78
79      public String getFuel() {
80          return fuel;
81      }
82
83      public void setFuel(String fuel) {
84          this.fuel = fuel;
85      }
86
87      public int getHp() {
88          return hp;
89      }
90
91      public void setHp(int hp) {
92          this.hp = hp;
93      }
94      public int getCc() {
95          return cc;
96      }
97
98      public void setCc(int cc) {
99          this.cc = cc;
100     }
101
102     public String getDescription() {
103         return description;
104     }
105
```

```
106     public void setDescription(String description)
107         this.description = description;
108     }
109
110     public int getClientId() {
111         return Client_Id;
112     }
113
114     public void setClientId(int Client_Id) {
115         this.Client_Id = Client_Id;
116     }
117 }
```

Listing B.23: VeiculosAdap class

```
1  public class VeiculosAdap  extends BaseAdapter {
2      private Context context;
3      private  int layout;
4      private ArrayList<Veiculos> veiculosArrayList;
5
6
7      public VeiculosAdap(Context context, int layout, ArrayList<↔
           Veiculos> veiculosArrayList) {
8          this.context = context;
9          this.layout = layout;
10         this.veiculosArrayList = veiculosArrayList;
11     }
12     @Override
13     public int getCount() {
14         return veiculosArrayList.size();
15     }
16
17     @Override
18     public Object getItem(int i) {
19         return veiculosArrayList.get(i);
20     }
21
22     @Override
23     public long getItemId(int i) {
24         return i;
25     }
26
27     private class ViewHolder{
28         TextView plate, brand, model, registration_year, color, fuel, ↔
               hp, cc, description, Client_Id;
29     }
30     @Override
31     public View getView(int position, View view, ViewGroup viewGroup) ↔
           {
32         View row = view;
33         ViewHolder holder = new ViewHolder();
34
35         if(row == null){
36             LayoutInflater inflater = (LayoutInflater) context.↔
                   getSystemService(Context.LAYOUT_INFLATER_SERVICE);
37             row = inflater.inflate(layout, null);
38
39             holder.plate= (TextView) row.findViewById(R.id.veh
AdapMat)↔
                   ;
```

```
40              holder.brand= (TextView) row.findViewById(R.id.vehAdapBr);
41              holder.model= (TextView) row.findViewById(R.id.vehAdapMod)↩
                    ;
42              holder.registration_year= (TextView) row.findViewById(R.id↩
                    .vehAdapReg);
43              holder.color= (TextView) row.findViewById(R.id.vehAdapCol)↩
                    ;
44              holder.fuel= (TextView) row.findViewById(R.id.vehAdapFu);
45              holder.hp= (TextView) row.findViewById(R.id.vehAdapHp);
46              holder.cc= (TextView) row.findViewById(R.id.vehAdapCc);
47              holder.description = (TextView) row.findViewById(R.id.↩
                    vehAdapDesc);
48              holder.Client_Id = (TextView) row.findViewById(R.id.↩
                    vehAdapID);
49              row.setTag(holder);
50          }
51          else {
52              holder = (VeiculosAdap.ViewHolder) row.getTag();
53          }
54
55          Veiculos veiculos = veiculosArrayList.get(position);
56
57          holder.plate.setText(veiculos.getPlate());
58          holder.brand.setText(veiculos.getBrand());
59          holder.model.setText(veiculos.getModel());
60          holder.registration_year.setText(veiculos.getRegistrationYear↩
                ());
61          holder.color.setText(veiculos.getColor());
62          holder.fuel.setText(veiculos.getFuel());
63          holder.hp.setText(veiculos.getHp());
64          holder.cc.setText(veiculos.getCc());
65          holder.description.setText(veiculos.getDescription());
66          holder.Client_Id.setText(veiculos.getClientId());
67
68          return row;
69      }
70  }
```

Listing B.24: VerMeusAnuncios class

```
1   public class VerMeusAnuncios extends Fragment {
2       View rootView;
3       SQLiteHelper sqLiteHelper;
4       ListView listView;    Array
5       List<Anuncios> list;
6       AnunciosAdap adapter = null;
7       private DatePickerDialog.OnDateSetListener dateSetListener;
8       private static final String TAG = "VerMeusAnuncios";
9
10      @Override
11      public View onCreateView(LayoutInflater inflater, ViewGroup ↩
            container,
12                               Bundle savedInstanceState) {
13          sqLiteHelper = new SQLiteHelper(getContext());
14          rootView = inflater.inflate(R.layout.fragment_view_anun,↩
15              container,false);
16
```

```
        listView = (ListView) rootView.findViewById(R.id.viewAnunList);
        list = new ArrayList<>();
        adapter = new AnunciosAdap(getActivity(), R.layout.fragment_anunadap,list);
        listView.setAdapter(adapter);
        showData();

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, final int position, long l) {

                CharSequence[] items = {"Editar", "Remover"};
                AlertDialog.Builder dialog = new AlertDialog.Builder(getActivity());


                dialog.setTitle("Selecione uma opcao");
                dialog.setItems(items, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int item) {

                        if (item == 0) {
                            // update
                            Cursor c = sqLiteHelper.query("SELECT " + sqLiteHelper.KEY_ID+" FROM " + sqLiteHelper.DB_TABLE);
                            ArrayList<Integer> arrID = new ArrayList<Integer>();
                            while (c.moveToNext()){
                                arrID.add(c.getInt(0));
                            }
                            // show dialog update at here
                            update(getActivity(), arrID.get(position));
                        } else {
                            // delete
                            Cursor c = sqLiteHelper.query("SELECT " + sqLiteHelper.KEY_ID+" FROM " + sqLiteHelper.DB_TABLE);
                            ArrayList<Integer> arrID = new ArrayList<Integer>();
                            while (c.moveToNext()){
                                arrID.add(c.getInt(0));
                            }
                            delete(String.valueOf(arrID.get(position));
                        }
                    }
                });
                dialog.show();
            }
        });

        return rootView;
    }

    public void showData(){
```

```
62          ArrayList<String> listData = new ArrayList<>();
63          Cursor cursor = sqLiteHelper.getAllData();
64          while (cursor.moveToNext())
65          {
66              listData.add(cursor.getString(cursor.getColumnIndex("title↩
                ")));
67          }
68          ListAdapter adapter = new ArrayAdapter<>(getActivity(), ↩
                android.R.layout.simple_list_item_1, listData);
69          listView.setAdapter(adapter);
70      }
71
72      public void delete(final String id) {
73          final AlertDialog.Builder dialogDelete = new AlertDialog.↩
                Builder(getActivity());
74          dialogDelete.setTitle("Atencao!!");
75          dialogDelete.setMessage("Anuncio selecionado sera removido!\↩
                n Deseja prossegair?");
76          dialogDelete.setPositiveButton("OK", new DialogInterface.↩
                OnClickListener() {
77              @Override
78              public void onClick(DialogInterface dialog, int which) {
79                  try {sqLiteHelper.deleteData(id
80                          );
81                          Toast.makeText(getActivity(), "Anuncio removido↩
                            com sucesso!", Toast.LENGTH_SHORT).show();

82                  } catch (Exception e){
83                      e.printStackTrace();
84                  }
85                  updateListView();
86              }
87          });
88
89          dialogDelete.setNegativeButton("Cancelar", new DialogInterface↩
                .OnClickListener() {
90              @Override
91              public void onClick(DialogInterface dialog, int which) {
92                  dialog.dismiss();
93              }
94          });
95          dialogDelete.show();
96      }
97
98      public void update(Activity activity, final int position) {
99
100         final Dialog dialog = new Dialog(activity);
101         dialog.setContentView(R.layout.fragment_update_anuncio);
102         dialog.setTitle("Update");
103
104         final EditText edtDate = (EditText) dialog.findViewById(R.id.↩
                txtDate);
105         final EditText edtTitle = (EditText) dialog.findViewById(R.id.↩
                txtTitle);
106         final EditText edtReward = (EditText) dialog.findViewById(R.id↩
                .txtReward);
107         final EditText edtLocatization = (EditText) dialog.↩
                findViewById(R.id.txtLocalization);
108         final EditText edtDescription = (EditText) dialog.findViewById↩
                (R.id.txtDescription);
```

```
final EditText edtVI = (EditText) dialog.findViewById(R.id.editVeID);

Button btnUpdate = (Button) dialog.findViewById(R.id.btnUpdate);
final TextView edtCD = (TextView) dialog.findViewById(R.id.editchooseDate);
// set width for dialog
int width = (int) (activity.getResources().getDisplayMetrics().widthPixels * 0.95);
// set height for  dialog
int height = (int) (activity.getResources().getDisplayMetrics().heightPixels * 0.97);
dialog.getWindow().setLayout(width, height);
dialog.show();

edtCD.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {

        int year = Calendar.YEAR;
        int month = Calendar.MONTH;
        int day = Calendar.DAY_OF_MONTH;

        DatePickerDialog dialog = new DatePickerDialog(
                getActivity(), android.R.style.Theme_Holo_Light_Dialog_MinWidth,
                dateSetListener,
                year, month, day);
        dialog.getWindow().setBackgroundDrawable(new
                ColorDrawable(Color.TRANSPARENT));

        dialog.show();
    }
});

dateSetListener = new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {
        month = month + 1;
        Log.d(TAG, year + "/" + month + "/" + day);

        String dateText = year + "/" + month + "/" + day;
        edtDate.setText(dateText);
    }
};

btnUpdate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            sqLiteHelper.updateData(String.valueOf(position),
                    edtDate.getText().toString().trim(),
                    edtTitle.getText().toString().trim(),
                    edtReward.getText().toString().trim(),
                    edtLocatization.getText().toString().trim(),
```

```
160                         edtDescription.getText().toString().trim()↩
161                         edtVI.getText().toString().trim()
162                         );
163                     dialog.dismiss();
164                     Toast.makeText(getActivity(), "Update feito com ↩
                           sucesso!",Toast.LENGTH_SHORT).show();
165                 }
166                 catch (Exception error) {
167                     Log.e("Update falhou!", error.getMessage());
168                 }
169                 updateListView();
170             }
171         });
172         String etdate = null, ettitle = new String(), etreward = null,↩
              etlocatization = null, etdescription = new String(), etVI ↩
             = new String();
173
174
175         try {   //for holding retrieve data from query and store in ↩
                the form of rows
176             Cursor cursor = sqLiteHelper.query("SELECT * " + " FROM "↩
                + sqLiteHelper.DB_TABLE + " WHERE " + sqLiteHelper.↩
                KEY_ID + " ='" + position + "'");
177
178             while (cursor.moveToNext()) {//Move the cursor to the next↩
                .
179                 etdate = cursor.getString(cursor.getColumnIndex("date"↩
                    ));
180                 ettitle = cursor.getString(cursor.getColumnIndex("title↩
                    "));
181                 etreward = cursor.getString(cursor.getColumnIndex("↩
                    reward"));
182                 etlocatization = cursor.getString(cursor.↩
                    getColumnIndex("localization"));
183                 etdescription = cursor.getString(cursor.getColumnIndex↩
                    ("description"));
184                 etVI = cursor.getString(cursor.getColumnIndex("↩
                    Vehicle_Id"));
185
186             }
187             edtDate.setText(etdate);
188             edtTitle.setText(ettitle);
189             edtReward.setText(etreward);
190             edtLocatization.setText(etlocatization);
191             edtDescription.setText(etdescription);
192             edtVI.setText(etVI);
193
194
195         } catch (Exception e) {
196             e.printStackTrace();
197         }
198         sqLiteHelper.close();
199         updateListView();
200     }
201
202     public void updateListView(){
203         // get all data from sqlite
204         ArrayList<String> listData = new ArrayList<>();
205         Cursor cursor = sqLiteHelper.getAllData();
```

```
206
207         list.clear();
208         while (cursor.moveToNext())
209         {
210             listData.add(cursor.getString(cursor.getColumnIndex("title↩
            }
211
212         ListAdapter adapter = new ArrayAdapter<>(getActivity(), ↩
               android.R.layout.simple_list_item_1, listData);
213         listView.setAdapter(adapter);
214     }
215
216
217 }
```

Listing B.25: VerMeusVeiculos class

```
1  public class VerMeusVeiculos extends Fragment {
2      View rootView;
3      Database database;
4      ListView listView;
5      public ArrayList<Veiculos> list;
6      VeiculosAdap adapter = null;
7      private DatePickerDialog.OnDateSetListener dateSetListener;
8      private static final String TAG = "VerMeusVeiculos";
9
10     @Override
11     public View onCreateView(LayoutInflater inflater, ViewGroup ↩
           container,
12                              Bundle savedInstanceState) {
13         database = new Database(getContext());
14         rootView = inflater.inflate(R.layout.fragment_view_veh,
15                 container, false);
16
17         listView = (ListView) rootView.findViewById(R.id.viewVeh);
18         list = new ArrayList<>();
19         adapter = new VeiculosAdap(getActivity(), R.layout.↩
               fragment_vehadap, list);
20         listView.setAdapter(adapter);
21         showData();
22
23         listView.setOnItemClickListener(new AdapterView.↩
               OnItemClickListener() {
24             @Override
25             public void onItemClick(final AdapterView<?> adapterView, ↩
                   View view, final int position, long l) {
26                 CharSequence[] items = {"Editar", "Remover"};
27                 AlertDialog.Builder dialog = new AlertDialog.Builder(↩
                       getActivity());
28
29                 dialog.setTitle("Selecione uma opcao");
30                 dialog.setItems(items, new DialogInterface.↩
                       OnClickListener() {
31                     @Override
32                     public void onClick(DialogInterface dialog, int ↩
                           item) {
33
34                         if (item == 0) {
35                             // update
```

```
36                          Cursor c = database.query("SELECT " + database.KEY_ID + " FROM " + database.DB_TABLE);
37                          ArrayList<Integer> arrID = new ArrayList<Integer>();
38                          while (c.moveToNext()) {
39                              arrID.add(c.getInt(0));
40                          }
41                          // show dialog update at here
42                          update(getActivity(), arrID.get(position));
43                      } else {
44                          // delete
45                          Cursor c = database.query("SELECT " + database.KEY_ID + " FROM " + database.DB_TABLE);
46                          ArrayList<Integer> arrID = new ArrayList<Integer>();
47                          while (c.moveToNext()) {
48                              arrID.add(c.getInt(0));
49                          }
50                          delete(String.valueOf(arrID.get(position)));
51                      }
52                  }
53              });
54              dialog.show();
55          }
56      });
57
58      return rootView;
59  }
60
61  public void showData() {
62      ArrayList<String> listData = new ArrayList<>();
63      Cursor cursor = database.getAllData();
64      while (cursor.moveToNext()) {
65
66          listData.add(cursor.getString(cursor.getColumnIndex("plate")));
67      }
68      ListAdapter adapter = new ArrayAdapter<>(getActivity(), android.R.layout.simple_list_item_1, listData);
69      listView.setAdapter(adapter);
70  }
71
72
73  public void delete(final String id) {
74      final AlertDialog.Builder dialogDelete = new AlertDialog.Builder(getActivity());
75      dialogDelete.setTitle("Atencao!!");
76      dialogDelete.setMessage("Veiculo selecionado sera removido!\nDeseja prosseguir?");
77      dialogDelete.setPositiveButton("OK", new DialogInterface.OnClickListener() {
78          @Override
79          public void onClick(DialogInterface dialog, int which) {
80              try { database.deleteData(id
81                  );
```

```
82                  Toast.makeText(getActivity(), "Veiculo removido ↔
                        com sucesso!", Toast.LENGTH_SHORT).show
                        ();
83              } catch (Exception e) {
84                  e.printStackTrace();
85              }
86              updateListView();
87          }
88      });
89
90      dialogDelete.setNegativeButton("Cancelar", new DialogInterface↔
            .OnClickListener() {
91          @Override
92          public void onClick(DialogInterface dialog, int which) {
93              dialog.dismiss();
94          }
95      });
96      dialogDelete.show();
97  }
98
99  public void update(Activity activity, final int position) {
100
101     final Dialog dialog = new Dialog(activity);
102     dialog.setContentView(R.layout.fragment_update_veiculo);
103     dialog.setTitle("Update");
104
105     final EditText edtPlate = (EditText) dialog.findViewById(R.id.↔
            editPlate);
106     final EditText edtBrand = (EditText) dialog.findViewById(R.id.↔
            editBrand);
107     final EditText edtModel = (EditText) dialog.findViewById(R.id.↔
            editModel);
108     final EditText edtAnR = (EditText) dialog.findViewById(R.id.↔
            editAnReg);
109     final EditText edtColor = (EditText) dialog.findViewById(R.id.↔
            editColor);
110     final EditText edtFuel = (EditText) dialog.findViewById(R.id.↔
            editFuel);
111     final EditText edtHP = (EditText) dialog.findViewById(R.id.↔
            editHP);
112     final EditText edtCc = (EditText) dialog.findViewById(R.id.↔
            editCC);
113     final EditText edtVCID = (EditText) dialog.findViewById(R.id.↔
            editVCID);
114     final EditText editDescription = (EditText) dialog.↔
            findViewById(R.id.editVDescription);
115
116     Button btnUpdate = (Button) dialog.findViewById(R.id.↔
            btnVupdate);
117     final TextView edtCD = (TextView) dialog.findViewById(R.id.↔
            editVCD);
118
119
120     // set width for dialog
121     int width = (int) (activity.getResources().getDisplayMetrics()↔
            .widthPixels * 0.95);
122     // set height for  dialog
123     int height = (int) (activity.getResources().getDisplayMetrics↔
            ().heightPixels * 0.97);
124     dialog.getWindow().setLayout(width, height);
125     dialog.show();
```

```
126
127         edtCD.setOnClickListener(new View.OnClickListener() {
128             @Override
129             public void onClick(View view) {
130                 int year = Calendar.YEAR;
131                 int month = Calendar.MONTH;
132                 int day = Calendar.DAY_OF_MONTH;
133
134                 DatePickerDialog dialog = new DatePickerDialog(
135                         getActivity(), android.R.style.↔
                            Theme_Holo_Light_Dialog_MinWidth,
136                         dateSetListener,
137                         year, month, day);
138                 dialog.getWindow().setBackgroundDrawable(new ↔
                    ColorDrawable(Color.TRANSPARENT));
139                 dialog.show();
140             }
141         });
142         dateSetListener = new DatePickerDialog.OnDateSetListener() {
143             @Override
144             public void onDateSet(DatePicker datePicker, int year, int←'
                    month, int day) {
145                 month = month + 1;
146                 Log.d(TAG, year + "/" + month + "/" + day);
147
148                 String dateText = year + "/" + month + "/" + day;
149                 edtAnR.setText(dateText);
150             }
151         };
152
153         btnUpdate.setOnClickListener(new View.OnClickListener() {
154             @Override
155             public void onClick(View v) {
156                 try        {
157                     database.update(
158                             String.valueOf(position),
159                             edtPlate.getText().toString().trim(),
160                             edtBrand.getText().toString().trim(),
161                             edtModel.getText().toString().trim(), edtAn
162                             R.getText().toString().trim(), edtColor.
163                             getText().toString().trim(), edtFuel.
164                             getText().toString().trim(),  edtHP.
165                             getText().toString().trim(),   edtCc.
166                             getText().toString().trim(),
167                             editDescription.getText().toString().trim↔
                                ()),
168                             edtVCID.getText().toString().trim());
169                     dialog.dismiss();
170                     Toast.makeText(getActivity(), "Update feito com ↔
                            Sucesso!", Toast.LENGTH_SHORT).show();
171                 } catch (Exception error) {
172                     Log.e("Update falhou!", error.getMessage());
173                 }
174                 updateListView();
175             }
176         });
177
```

```
178         String edPlate = null, edBrand = new String(), ed
               Model = null,↔edAnR = null, edColor = new String
               (), edFuel = new String↔(), edHP = null, edCc =
               null, edescription = null, edVCID =↔new String();
179
180         try {   //for holding retrieve data from query and store in ↔
               the form of rows
181             Cursor cursor = database.query("SELECT *" + " FROM "+ ↔
                   database.DB_TABLE + " WHERE " + database.KEY_ID
                   + " ='"↔
                    + position + "'");
182
183
184             while (cursor.moveToNext()) {//Move the cursor to the next↔
                    row.
185                 edPlate = cursor.getString(cursor.getColumnIndex("↔
                       plate"));
186                 edBrand = cursor.getString(cursor.getColumnIndex("↔
                       brand"));
187                 edModel = cursor.getString(cursor.getColumnIndex("↔
                       model"));
188                 edAnR = cursor.getString(cursor.getColumnIndex("↔
                       registration_year"));
189                 edColor = cursor.getString(cursor.getColumnIndex("↔
                       color"));
190                 edFuel = cursor.getString(cursor.getColumnIndex("fuel"↔
                       ));
191                 edHP = cursor.getString(cursor.getColumnIndex("hp"));
192                 edCc = cursor.getString(cursor.getColumnIndex("cc"));
193                 edescription = cursor.getString(cursor.getColumnIndex(↔
                       "description"));
194                 edVCID = cursor.getString(cursor.getColumnIndex("↔
                       Client_Id"));
195             }
196             edtPlate.setText(edPlate);
197             edtBrand.setText(edBrand);
198             edtModel.setText(edModel);
199             edtAnR.setText(edAnR);
200             edtColor.setText(edColor);
201             edtFuel.setText(edFuel);
202             edtHP.setText(edHP);
203             edtCc.setText(edCc);
204             editDescription.setText(edescription);
205             edtVCID.setText(edVCID);
206
207
208         } catch (Exception e) {
209             e.printStackTrace();
210         }
211         database.close();
212         updateListView();
213
214     }
215
216     private void updateListView() {
217         // get all data from sqlite
218         ArrayList<String> listData = new ArrayList<>();
219         Cursor cursor = database.getAllData();
220
221         list.clear();
222         while (cursor.moveToNext()) {
```

```
223
224             listData.add(cursor.getString(cursor.getColumnIndex("p"))); late↩
            }
225
226         ListAdapter adapter = new ArrayAdapter<>(getActivity(), ↩
                android.R.layout.simple_list_item_1, listData
227         listView.setAdapter(adapter);
228     }
229
230 }
```

Listing B.26: DatabaseHelper class

```
1  public class DatabaseHelper extends SQLiteOpenHelper{
2
3      private static final int DATABASE_VERSION = 1;
4      private static final String DATABASE_NAME = "fot.sqlite";
5      private static final String DB_TABLE = "FOTOS";
6      private static final String KEY_ID = "id";
7      private static final String KEY_IMAGE = "image";
8
9      private static final String TABLE_SEND_PHOTOS = "RECIVED";
10     private static final String RECIVED_PHOTO_ID = "id_rp";
11     private static final String RECIVED_PHOTO_NAME = "pic";
12
13     // Table create statement
14     private static final String CREATE_TABLE_IMAGE = "CREATE TABLE IF ↩
           NOT EXISTS" + DB_TABLE + "("+
15             KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
16             KEY_IMAGE + " BLOB);";
17
18
19     public DatabaseHelper(Context context) {
20         super(context, DATABASE_NAME, null, DATABASE_VERSION);
21
22     }
23
24     public void queryData(String sql){
25         SQLiteDatabase database = getWritableDatabase();
26         database.execSQL(sql);
27     }
28
29     @Override
30     public void onCreate(SQLiteDatabase sqLiteDatabase) {
31         sqLiteDatabase.execSQL(CREATE_TABLE_IMAGE);
32         sqLiteDatabase.execSQL(RECIVED_TABLE);
33
34     }
35
36     @Override
37     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1↩
           ) {
38         sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
39         onCreate(sqLiteDatabase);
40
41     }
42
43     public void insertData(byte[] image) throws SQLiteException {
44         SQLiteDatabase database = this.getWritableDatabase();
```

```
45          String sql = "INSERT INTO " + DB_TABLE +"(id,image) VALUES
                ?)";                    (?,↩
46
47          ContentValues values = new C
48          values.put(KEY_IMAGE,image);ontentValues();
49          database.insert(DB_TABLE, nu--     ,   `
50
51      }
52
53      //send
54      pr ivate static final String RECI VED_TABLE = " CREATE TABLE IF NOT ↩
            EXISTS "+TA BLE_SEND_PH OTOS + "("+
55              RECIVED_PHOTO_ID + " INT EGER PRIMARY KEY AUTOINCREMENT," +
56              RECIVED_PHOTO_NAME + " B LOB);";
57      public void getData(byte[] pic) throws SQLiteException {
58          SQLiteDatabase database = th is.getWritableDatabase();
59          String sql = "INSERT INTO " + TABLE_SEND_PHOTOS +"(id_rp,pic)
                VALUES (?, ?)";         ↩
60
61          ContentValues values = new C ontentValues
62          values.put(RECIVED_PHOTO_NAM(); E,pic);
63          database.insert(TABLE_SEND_P HOTOS, null, values);
64
65          // SQLiteStatement statement  - database.compileStatement(
66          /* statement.bindBlob(1,pic);
67          statement.executeInsert();*/
68
69      }
70  }
```