LiPRIF
Aplicativo para identificação de permissão de acesso de veículos e condutores ao estacionamento do IFRS

Trabalho de Conclusão do Curso de
Tecnologia em Sistemas para Internet

Marcos Vinicius da Silva Rogowski
Orientador: Marcelo A. Rauh Schmitt

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Porto Alegre
Av. Cel. Vicente, 281, Porto Alegre - RS - Brasil

mvrogowski@gmail.com, marcelo.schmitt@poa.ifrs.edu.br

Resumo. *Este artigo apresenta uma proposta de sistema para controle de acesso de veículos a estacionamentos baseado em reconhecimento de placas. Trata-se de uma solução gratuita, realizada através de um aplicativo mobile para a plataforma Android que permite ao responsável pela permissão de entrada identificar os veículos e seus condutores. O aplicativo, através da câmera do smartphone, identifica a placa do veículo, consulta o cadastro com as permissões e sinaliza ao segurança se o motorista e seu veículo estão ou não autorizados a ingressar no estacionamento. As tecnologias utilizadas são: OCR (Optical Character Recognition) para identificação das placas; Java para desenvolvimento de aplicações para plataforma Android; Javascript para o desenvolvimento do painel administrativo; Python e MongoDB para o desenvolvimento do Web Service.*

1. Introdução

O campus Porto Alegre do IFRS apresenta um problema que é compartilhado por outras instituições: a falta de recursos para implementar um sistema de cancelas. Atualmente, para controlar o acesso ao estacionamento, é necessário que um dos seguranças do campus faça a identificação do veículo através de um selo. Esse selo tem diversas cores e legendas, permitindo que o segurança identifique se o carro é de um aluno, servidor ou convidado e também a validade do selo para o semestre corrente.

Esta abordagem traz problemas burocráticos, logísticos e de segurança: é necessário um novo edital para confecção de novos selos - o que gera um custo recorrente ao campus e um longo prazo de entrega, fazendo com que, caso os selos não sejam entregues em tempo hábil, seja necessário permitir o acesso ao estacionamento com selos antigos ou comprovantes de matrícula; distribuir os selos aos seus usuários - há casos em que o selo é extraviado, obrigando o usuário a solicitar um novo selo; se o usuário tem mais de um veículo, necessita mais de um selo; ao final do semestre, o recolhimento dos selos não é efetuado - permanecendo com o selo em seu veículo, o usuário poderá acessar as dependências da instituição mesmo se já tiver sido desligado da mesma; no caso dos alunos que tem seu acesso restrito ao turno em que tem aulas, quando as aulas ocorrem em mais de um turno - manhã e noite, por exemplo - necessita-se a utilização de dois selos.

Atualmente, encontram-se diversas opções para automação de cancelas para controle de acesso: cancelas com tíquetes, cancelas com cartões ópticos e magnéticos, selos com RFID (*Radio Frequency Identification*), identificação por câmeras e identificação manual com documentação. Porém, conforme apresentado na seção 2, todas as soluções supra mencionadas são onerosas, pois necessitam aquisição de equipamentos ou contratação de serviços.

Uma solução viável é desenvolver um aplicativo para *smartphones* no qual o segurança responsável pelo controle de acesso ao estacionamento fará a leitura da placa do veículo - utilizando uma técnica de leitura automatizada de placas de licenciamento, o aplicativo fará a identificação das mesmas e após consultará no sistema as permissões de acesso, aferindo e informando na tela do *smartphone* a permissão de acesso e permanência do veículo bem como do motorista que o está conduzindo.

Através da solução apresentada neste artigo, pretende-se reduzir os custos logísticos e operacionais intrínsecos da confecção, gerenciamento e distribuição dos selos; e pretende-se aumentar a segurança do acesso dos veículos ao estacionamento, dificultando que usuários não autorizados acessem as dependências do campus.

Tendo em vista o problema apresentado, tem-se como objetivo desenvolver um sistema que não exija a compra de equipamentos e selos; que permita o controle eficiente por parte da segurança; que dê agilidade ao registro e cancelamento de veículos; e que possa ser utilizado em outros campi do IFRS ou mesmo outras organizações.

O seção 2 apresenta trabalhos relacionados; a seção 3 traz o conceito da tecnologia aplicada na solução; a seção 4, mostra a metodologia utilizada para desenvolvimento da solução e cronograma; a seção 5 demonstra como a solução foi desenvolvida; a seção 6 descreve os resultados da validação das aplicações; a seção 7 enumera algumas melhorias que podem ser efetuadas no projeto; e, por último, são apresentadas as considerações finais na seção 8.

2. Trabalhos relacionados

Esta seção faz referência a algumas empresas atuantes no mercado que fornecem soluções para controle de acesso a estacionamentos.

2.1. Soluções pesquisadas

Passo Automação[1] é uma empresa situada na cidade de Canoas/RS que atua no mercado de controle de acesso desde o ano de 1990. Entre seus produtos oferecidos, existem as cancelas automatizadas para estacionamentos. Para cada cancela, existe um dispensador ou validador com leitor óptico de tíquetes - estas podem ser adaptadas para efetuar leitura de cartões ópticos, magnéticos e RFID. A Figura 1 mostra alguns de seus produtos já instalados.

[1] Site da Passo Automação: www.passo.com.br

Figura 1. *Totens* e cancelas de estacionamento fornecidas por Passo Automação.

Henry[2] é uma empresa fundada em 1995 atuando, inicialmente, como fornecedora de soluções para controle de pontos. Posteriormente, começou a atuar no segmento de controle de acesso, tendo um produto focado em estacionamentos, também - a Figura 2 mostra exemplos de alguns de seus totens.

Figura 2. Totens e cancelas de estacionamento fornecidas por Henry.

[2]Site da Henry: www.henry.com.br

A Embratecc[3] é uma empresa situada no Rio de Janeiro, fundada em 2008, cuja atuação é exclusivamente no setor de automação de estacionamentos - a Figura 3 mostra exemplos de alguns de seus totens.

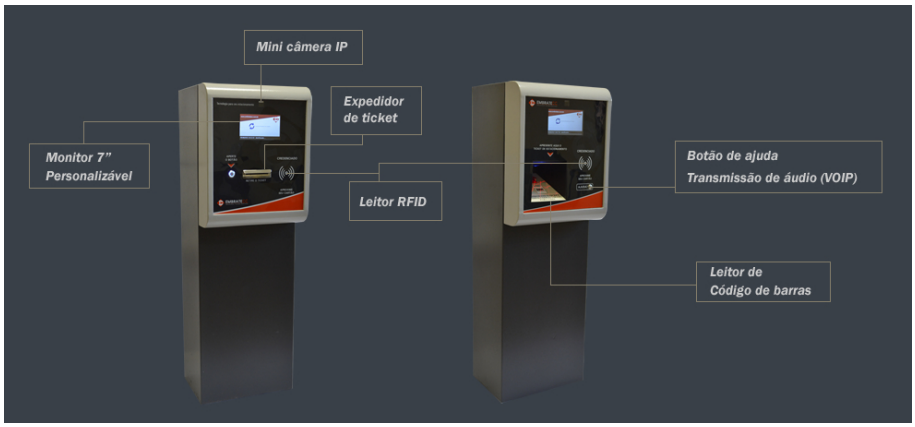


Figura 3. Totens de estacionamento fornecidas por Embratecc.

2.2. Comparação entre as soluções

Para cada uma das empresas supra referidas foi solicitado um orçamento. Para garantir que os orçamentos fossem baseados nas mesmas premissas, a única descrição fornecida para as empresas foi: ”uma faculdade pública com estacionamento isento para servidores e alunos com uma entrada e uma saída de veículos”.

O Quadro 1 ilustra uma síntese da comparação entre os orçamentos fornecidos pelas empresas mencionadas:

Itens	Passo	Henry	Embratecc
Tíquete	X	-	X
RFID	-	X	X
Leitor óptico	X	X	X
Software de gestão	X	X	X
Compra da solução	R$ 45.200,00	R$ 34.086,00	R$ 49.800,45
Aluguel mensal da solução	R$ 3.050,00	-	R$ 3.000,00

Quadro 1. Diferenças entre as soluções.

Pode ser visto no Quadro 1, como as empresas pesquisadas já atuam há bastante tempo e estão consolidadas no mercado, os serviços prestados são similares - com uma pequena divergência de preços. Todas elas oferecem a opção de aquisição de seus equipamentos, porém somente a Passo Automação e a Henry ofereceram a opção de contratação do serviço através de locação mensal dos equipamentos.

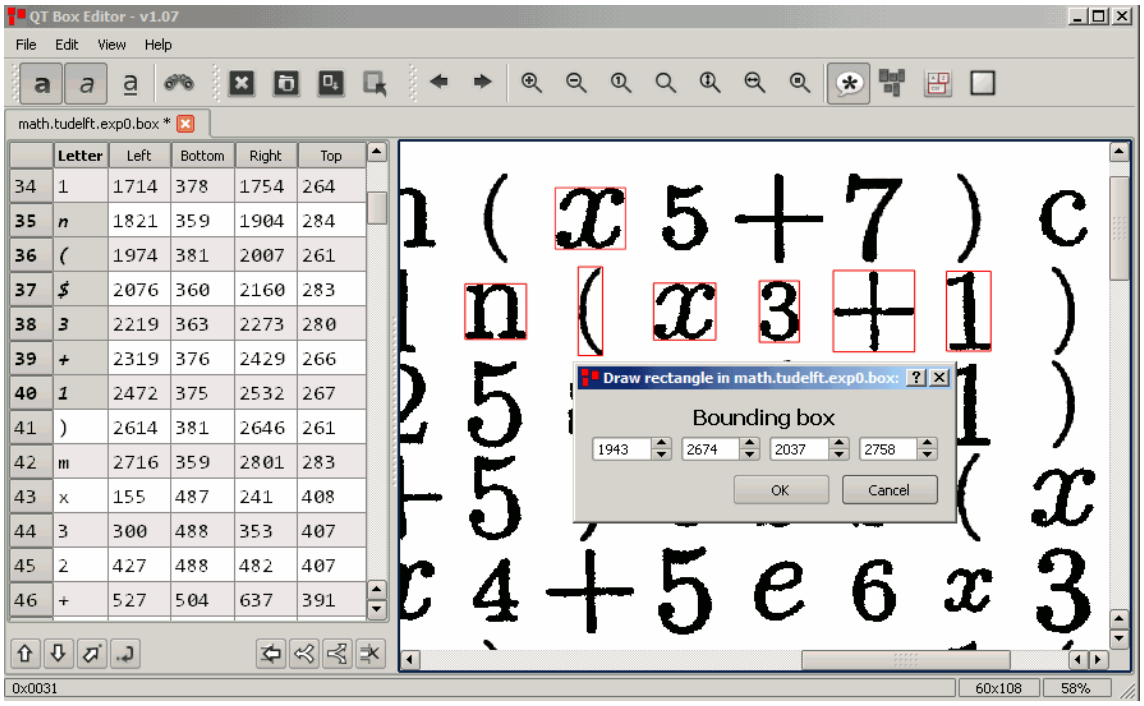
Embora diversas empresas ofereçam o serviço de controle de acesso através de câmeras - dentre as que enviaram orçamento, de acordo com os seus sites, apenas a Embratecc oferece esse serviço -, nenhuma delas disponibiliza o serviço utilizando plataformas móveis para fazer a operacionalização.

[3]Site da Embratecc: www.embratecc.com.br

3. Reconhecimento Óptico de Caracteres

De acordo com [Eikvil 1993], OCR - *Optical Character Recognition* - é basicamente um reconhecimento de padrões. No caso dos OCRs, os padrões a serem encontrados são caracteres de texto, números, caracteres de pontuação, etc.

O primeiro passo para o reconhecimento de caracteres é ensinar a máquina quais os padrões podem aparecer e como eles se parecem. Segundo [Zareba 2016], o ensinamento da máquina é efetuado através da submissão de imagens dos diferentes tipos de caracteres, tipos de fontes e todas as outras classes de caracteres que seriam alvo do reconhecimento automatizado juntamente com um arquivo contendo o mapeamento dos caracteres - o arquivo de mapeamento consiste em um arquivo de texto contendo as regiões onde cada caractere é encontrado e qual caractere ele representa. A Figura 4 ilustra o a criação do arquivo de mapeamento dos caracteres através da ferramenta *Qt Box Editor*[4].



	Letter	Left	Bottom	Right	Top
34	1	1714	378	1754	264
35	n	1821	359	1904	284
36	(	1974	381	2007	261
37	$	2076	360	2160	283
38	3	2219	363	2273	280
39	+	2319	376	2429	266
40	1	2472	375	2532	267
41	)	2614	381	2646	261
42	m	2716	359	2801	283
43	x	155	487	241	408
44	3	300	488	353	407
45	2	427	488	482	407
46	+	527	504	637	391

Figura 4. Criando arquivo de mapeamento de caracteres utilizando a ferramenta *Box Editor* [Zareba 2016]

Após o treinamento da ferramenta de OCR, para efetuar o reconhecimento dos caracteres, os seguintes passos devem ser executados, conforme ilustra a Figura 5:

1. Escaneamento óptico: através de um processo de escaneamento é capturada uma imagem digital de um documento [Eikvil 1993];
2. Localização e segmentação: de acordo com [Chen et al. 2000], o ponto-chave de um algoritmo de segmentação é encontrar uma maneira de diferenciar os *pixels* do texto dos *pixels* do fundo;
3. Pré-processamento: conforme [Eikvil 1993], esta etapa é responsável por eliminar ruídos provenientes da digitalização do documento alvo; e, de acordo com

[4]Site do Qt Box Editor: http://zdenop.github.io/qt-box-editor/

[Cheriet et al. 2007], esta etapa envolve os processos de suavização e remoção de ruídos, análise do grau de enviesamento e correção, análise de inclinação, normalização do caractere, análise/definição de contorno e afinamento.

4. Extração de características: segundo [Eikvil 1993], nessa etapa são extraídas as características essenciais dos símbolos, e essa é considerada uma das partes mais difíceis do reconhecimento de padrões.
5. Classificação: aqui os símbolos são associados às classes de caracteres conhecidas [Eikvil 1993].
6. Pós processamento: aqui, na etapa de ”Agrupamento”, os símbolos são agrupados, formando palavras ou números; após, existe a etapa de ”Detecção de Erros e Correção”, as palavras são ajustadas considerando o contexto [Eikvil 1993].

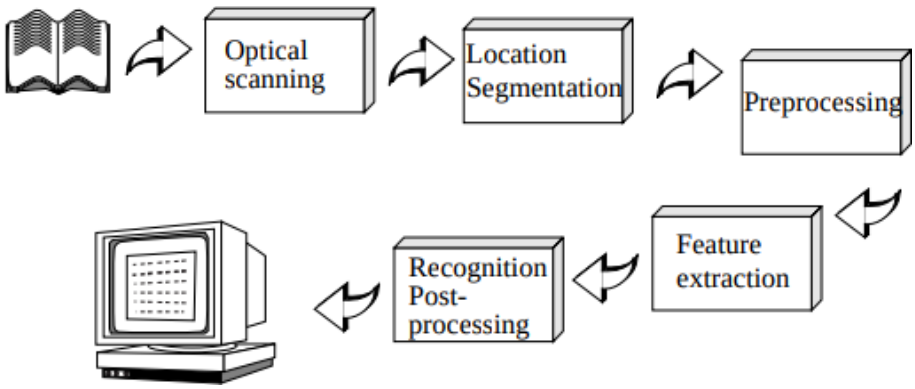


Figura 5. Etapas de um OCR. [Eikvil 1993]

Para extrair o texto das imagens, primeiramente, é necessário digitalizar a imagem através de um escâner óptico - uma câmera, por exemplo. Após a localização das regiões do documento que contém textos, cada símbolo é extraído através de um processo de segmentação e são pré-processados para eliminar ruídos, facilitando a extração de características de cada símbolo.

A identificação de cada caractere é feita comparando as características obtidas com a descrição dos símbolos previamente ensinados à máquina. Em seguida, são utilizadas informações contextuais para reconstruir as palavras e números do documento original.

No caso do objeto deste artigo, a câmera do *smartphone* digitalizará a imagem de uma placa de licenciamento de veículo e extraíra o texto nela contido para fazer uma busca na base de dados e aferir a permissão de acesso ao estacionamento da instituição.

4. Metodologia

A metodologia utilizada para desenvolvimento da solução é a seguinte:

1. Definição das funcionalidades: foram levantados os requisitos mínimos para definição do escopo do MVP (*Minimal Viable Product*).
2. *Benchmarking*: foi feito um estudo de estado da arte e analisadas três soluções de controle de acesso.

3. Requisitos tecnológicos: foi realizada uma pesquisa sobre as tecnologias necessárias para implementação.
4. Pesquisa bibliográfica: foi efetuada uma pesquisa de fontes confiáveis para fundamentar as técnicas aplicadas.
5. Desenvolvimento da aplicação: a aplicação foi desenvolvida a partir da leitura automatizada das placas utilizando uma abordagem ágil e incremental.
6. Validação da solução: após a implementação, foi realizado teste de campo inicial no estacionamento do IFRS. A partir dos resultados dos testes, serão implementadas as correções necessárias. Após os ajustes, os seguranças testarão a usabilidade do aplicativo.

5. Desenvolvimento da aplicação

Para implementação da solução, após o levantamento de requisitos da aplicação, modelou-se o MVP, cujos códigos fontes de todo o projeto estão disponíveis no BitBucket[5], que é composto pelos seguintes módulos:

- Aplicativo para a plataforma *Android* - a aplicação efetua a leitura da placa do veículo utilizando a câmera do *smartphone* e, através de um OCR implementado pela biblioteca OpenALPR[6], identifica a placa e consulta os dados cadastrais do veículo e de seu condutor, permitindo ao segurança validar as permissões de entrada no estacionamento;
- Painel administrativo - este sistema disponibiliza uma interface para o gerenciamento dos usuários do estacionamento;
- *Web Service* - essa aplicação tem o objetivo de disponibilizar e persistir os dados gerados pelo aplicativo Android e pelo painel administrativo.

5.1. Aplicativo *Android*

O aplicativo serve como ferramenta para o segurança autorizar ou não a entrada de veículos no estacionamento. Conforme ilustra a Figura 6, ao executar o aplicativo, o segurança pode inicializar o processo de leitura automatizada de placas ou a consulta manual da placa, neste caso, a partir da digitação a placa do veículo.

[5]Repositórios do projeto: https://bitbucket.org/liprif/
[6]OpenALPR: www.openalpr.com

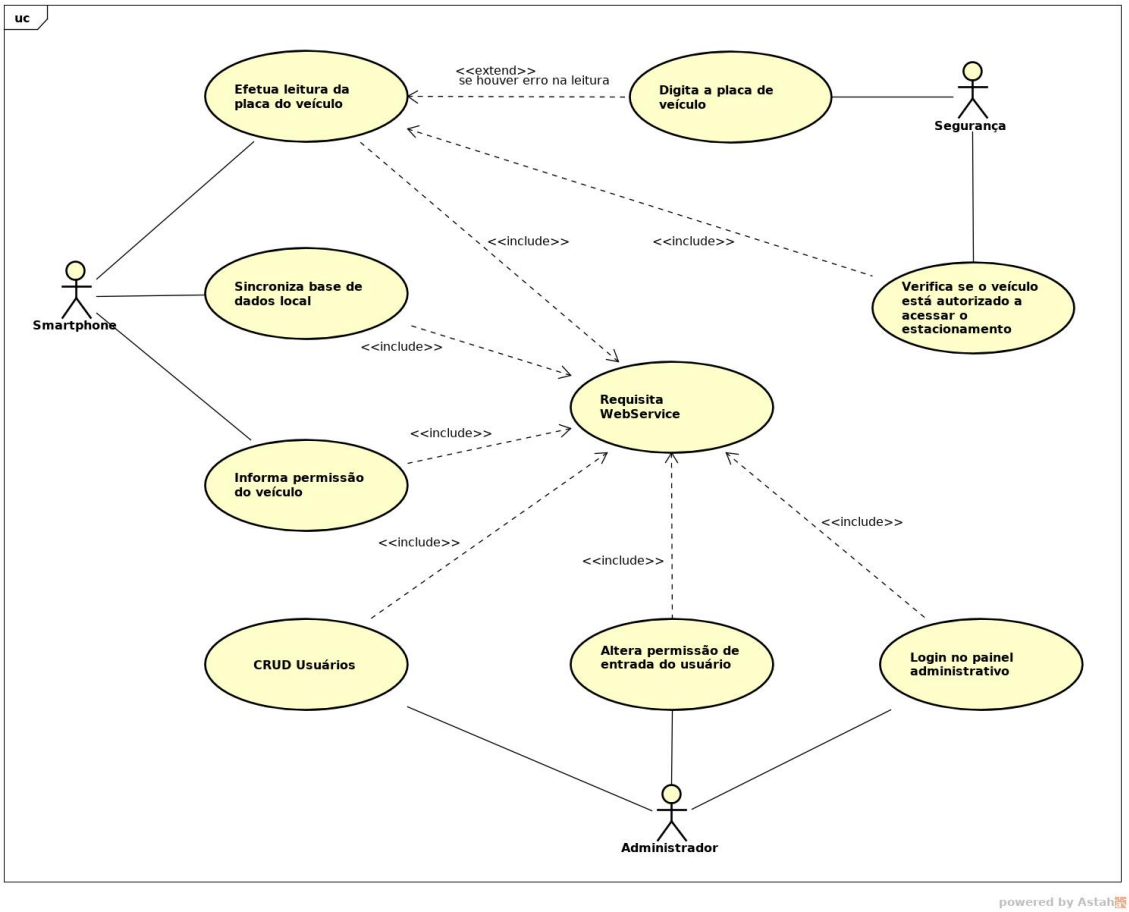
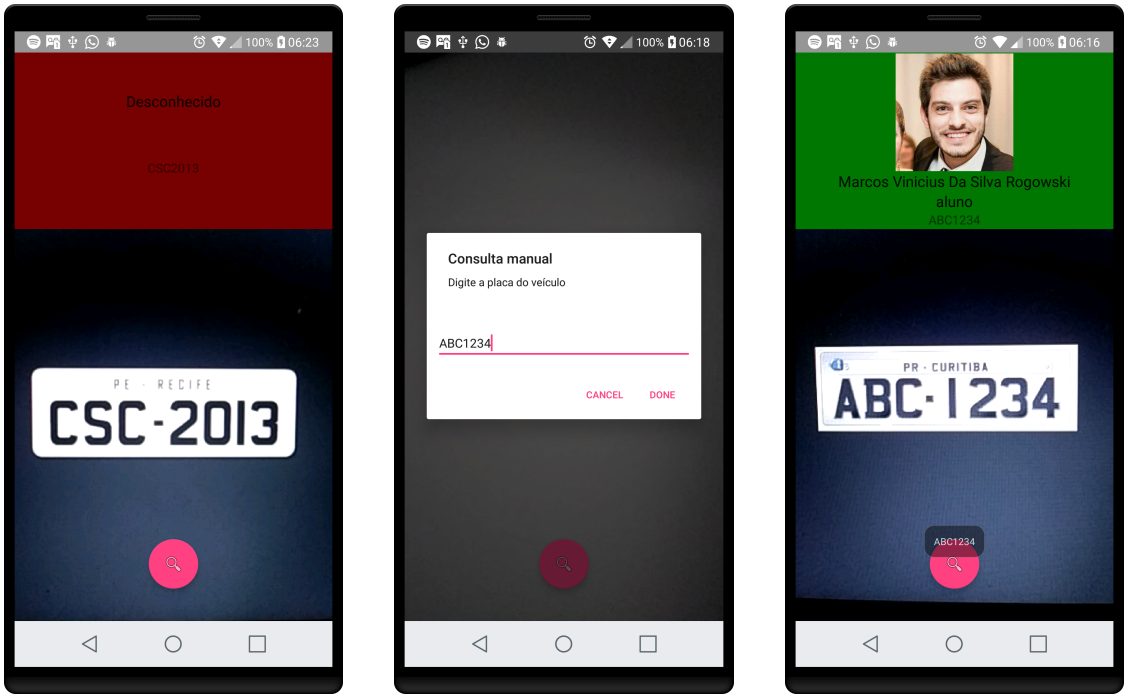


Figura 6. Diagrama de casos de uso do aplicativo.

No modo de leitura automatizada, após inicializada a câmera, o segurança deverá apontá-la para a placa do veículo. Caso ocorra qualquer erro na leitura automatizada ou o veículo não esteja cadastrado, o aplicativo informará na tela que o veículo não foi encontrado - conforme ilustra a Figura 7(a) -, e o segurança poderá aferir se a placa foi identificada corretamente. Caso não tenha sido corretamente identificada, ele poderá efetuar a consulta manualmente, conforme ilustra a Figura 7(b). Caso o veículo tenha sido identificado, tanto manual quando automaticamente, o aplicativo informa ao segurança que o condutor está autorizado a entrar, conforme mostra a Figura 7(c).

5.1.1. Interface

Tendo como proposta a simplificação do processo de identificação de veículos, optou-se por desenvolver uma interface com poucos componentes para ter fluidez e facilitar a operação do aplicativo. Logo, optou-se por utilizar Fragmentos para manipular sua interface, pois troca de contexto e a manipulação de Atividades é mais onerosa ao processador do *smartphones* que os Fragmentos. Conforme [Andoid Developers 2018a], um Fragmento representa o comportamento ou uma parte da interface do usuário em um Atividade; é possível combinar vários fragmentos em uma única atividade para compilar uma interface de usuário e reutilizar um fragmento em diversas atividades. Um Frag-



(a) Tela para usuário negado.

(b) Tela para consulta manual da placa.

(c) Tela para usuário liberado.

Figura 7. Telas do aplicativo Android.

mento é como uma seção modular de uma atividade, que tem o próprio ciclo de vida, recebe os próprios eventos de entrada e que pode ser adicionado ou removido com a atividade em execução (uma espécie de ”subatividade”reutilizável). Atividades, conforme [Andoid Developers 2018b], é o ponto de entrada de qualquer interação do usuário com o aplicativo. Cada uma delas disponibiliza uma janela para renderização da interface de usuário e podem invocar outras atividades para executar diferentes ações.

5.1.2. Identificação automatizada

Para identificação automatizada da placa, é utilizada uma biblioteca de código aberto chamada OpenALPR - a sigla ALPR significa *Automatic License Plate Recognizer*. Essa biblioteca é responsável por executar a função do OCR, extraindo o texto das placas a partir de uma imagem. Uma das principais vantagens de se utilizar essa biblioteca é que a ela já passou pela etapa de aprendizagem, sendo otimizada para a leitura de placas de licenciamento de veículos, como o próprio nome sugere.

A biblioteca OpenALPR tem a capacidade de extrair o texto de imagens estáticas bem como de *streams* de imagens - como o *preview* da câmera antes de obturar a imagem, por exemplo. Visando agilizar o processo de identificação, optou-se por utilizar a segunda forma. Infelizmente, o Android não conta com um componente nativo e flexível para manipulação dos *streams*. Para isso, é necessário implementar um componente personalizado. Para implementar essa funcionalidade, utilizou-se como referência uma biblioteca chamada react-native-openalpr, cuja publicação foi feita por [Corcos 2017]. Essa biblio-

teca foi escrita para o *framework* de desenvolvimento mobile chamado de React Native[7] e utiliza as bibliotecas OpenCV[8] - biblioteca de código aberto para computação visual - e a já comentada OpenALPR.

O *framework* React Native permite que aplicações para plataformas móveis sejam escritas utilizando a linguagem de programação Javascript. A ideia desse *framework* é criar uma abstração para os componentes nativos das plataformas alvo. Uma funcionalidade muito interessante dele é poder implementar a solução nativamente - se o desenvolvedor precisar utilizar um recurso nativo da plataforma e não exista um módulo para isso ou reutilizar qualquer modulo já implementado nativamente, como um código de alta performance, por exemplo -, e utilizar um *Bridge Pattern* para fazer a ligação do código Javascript com o código nativo. Logo, para fazer a adaptação da biblioteca react-native-openalpr, reutilizando o código nativo do componente de captura e reconhecimento, foi necessário apenas adicionar o código nativo da biblioteca ao projeto e ajustar as interfaces de acesso aos seus métodos, conforme pode ser visto no código fonte.

5.1.3. Sincronização dos dados

Após identificada a placa, o aplicativo consulta no cadastro se o veículo está autorizado a ingressar no estacionamento e quem é seu condutor. Para garantir que a identificação possa ser efetuada mesmo quando há indisponibilidades na rede, o aplicativo, através de uma rotina, que é executada em segundo plano, faz a sincronização do banco de dados remoto com um banco de dados local - a plataforma *Android* disponibiliza um bancos de dados chamado de SQLite para armazenamento de dados locais. Essa rotina é executada na inicialização, em intervalos de vinte minutos e toda a vez que uma placa não é encontrada na base local, garantindo que uma placa inserida no intervalo entre as sincronizações não seja ignorada.

5.2. Painel administrativo

Para o desenvolvimento da interface administrativa, foi utilizado um *framework* chamado react-admin[9], baseado em *Material Design*[10] e construído utilizando a biblioteca para a linguagem Javascript chamada React[11].

Esse sistema permite efetuar operações de gerenciamento dos usuários com permissão de acesso ao estacionamento. Através dele é possível: criar, editar ou remover um usuário; listar os motoristas cadastrados, efetuar buscas por placa e/ou nome; alterar a permissão de entrada do veículo.

Primeiramente, é necessário efetuar o login na plataforma, conforme ilustra a Figura 8, utilizando um usuário com permissões de administrador. Para o MVP, têm-se apenas um usuário com poder administrativo, chamado ”admin”.

[7]React native: https://facebook.github.io/react-native/
[8]OpenCV: https://opencv.org/
[9]React admin: https://marmelab.com/react-admin/
[10]Material Design: https://material.io/
[11]React: https://reactjs.org/

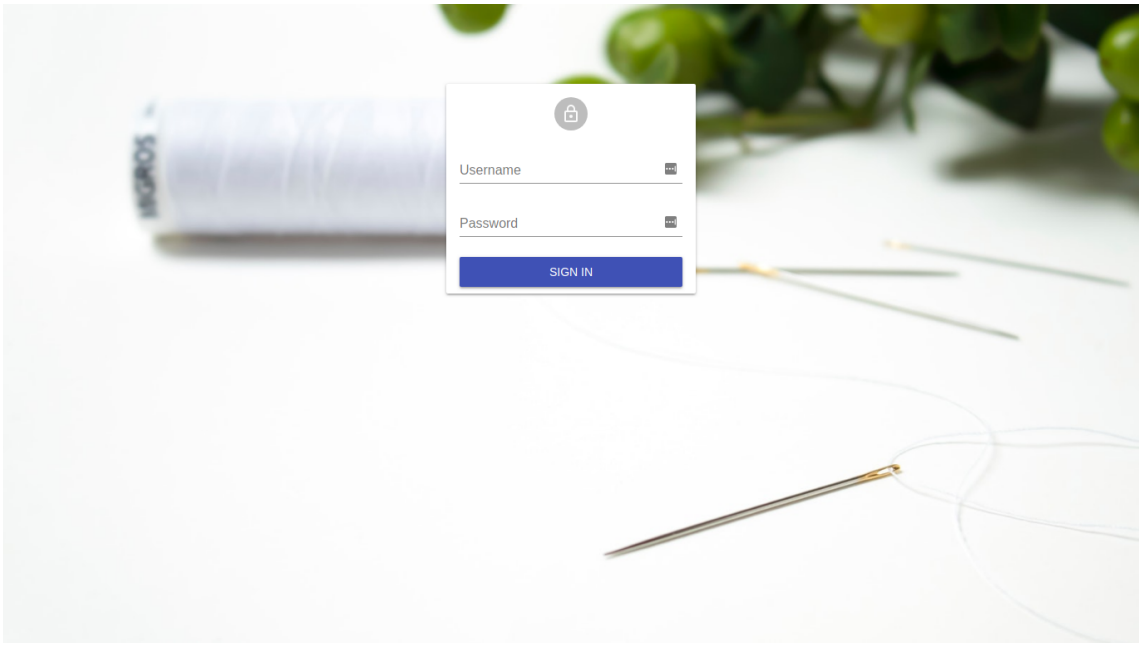


Figura 8. Tela de login.

Após o login, os usuários cadastrados na plataforma são listados numa tabela, conforme mostra a Figura 9, onde é possível efetuar as ações de busca, edição, remoção, adição de motoristas, bem como alterar as permissões de acesso deles. Todos os campos da lista podem ser ordenados para facilitar a consulta dos dados.

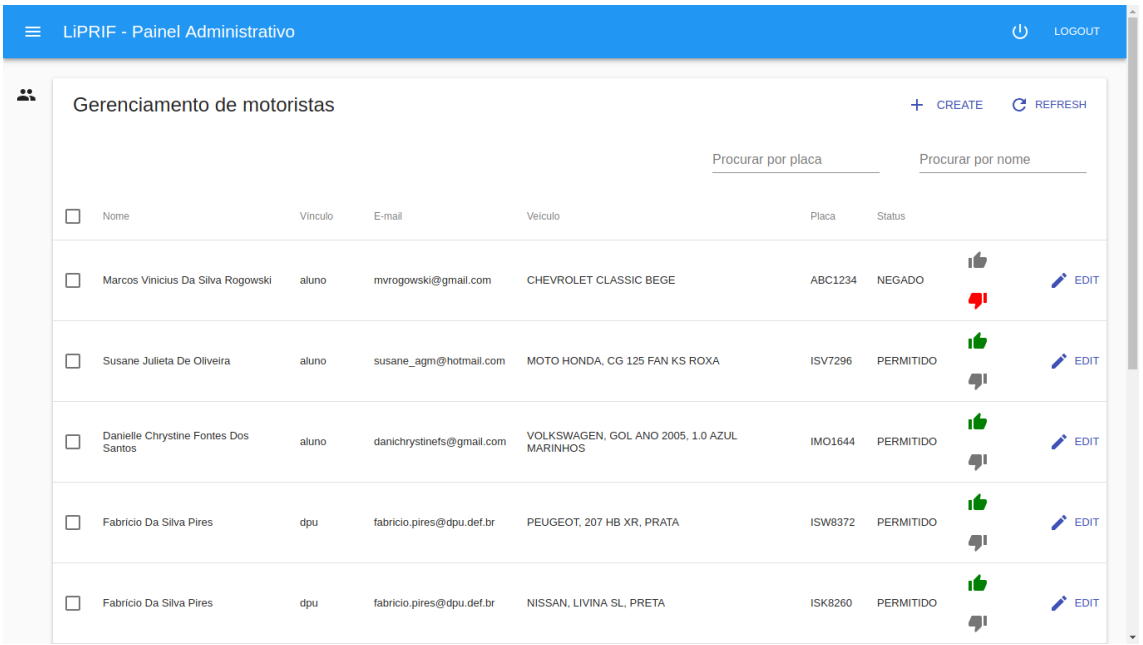


Figura 9. Lista dos motoristas cadastrados.

5.3. Web Service

Para servir como *backend* para o painel administrativo e para o aplicativo Android, foi desenvolvido um *web service* com arquitetura RESTful escrito em Python, cujos endereços podem ser vistos no Quadro 2. Para a persistência dos dados foi utilizado um banco de dados não relacional (NoSQL) orientado à documentos, onde cada documento é representado por um objeto JSON (*Javascript Object Notation*), chamado MongoDB[12]. O Bloco 1 ilustra a estrutura básica dos documentos utilizados no projeto.

```
{
        "_id" : ObjectId("5b05c2ecf8d3a7147d0fee7b"),
        "name" : "Marcos Vinicius Da Silva Rogowski",
        "email" : "mvrogowski@gmail.com",
        "phones" : ["51981792244"],
        "vehicle" : "CHEVROLET CLASSIC BEGE",
        "plate" : "ABC1234",
        "status" : "NEGADO",
        "reg_number" : "1004123",
        "bond" : "aluno",
        "role" : ["user"]
}
```

Bloco 1. Estrutura básica de um documento.

Método HTTP	Endereço	Descrição
GET	/drivers	Retorna a lista de usuários
GET	/drivers/{id}	Retorna um usuário específico
POST	/drivers	Cria um novo usuário motorista
DELETE	/drivers/{id}	Remove um usuário
PUT	/drivers/{id}	Atualiza informações do usuário
POST	/oauth2/token	Obtém *token* de acesso à API
POST	/oauth2/introspect	Verifica se o *token* é válido

Quadro 2. Sumário dos endereços da API.

As informações dos usuários contidas na base de dados foram fornecidas, através de uma planilha de Excel, pelo setor do campus responsável pelo cadastro de usuários do estacionamento. Para extrair os dados da planilha e popular a base de dados, foi escrito um pequeno *script* em Python.

6. Validação

A validação do aplicativo foi efetuada fazendo a identificação de vinte veículos dentro do estacionamento do IFRS durante a noite - foi escolhido o turno da noite por ser considerado o ambiente de operação mais hostil devido a baixa luminosidade. A Tabela 3 mostra o resultado do teste.

[12]Site MongoDB: https://www.mongodb.com/

Placa	Reconhecimento	Tempo	Permissão
A*****9	AUTOMÁTICO	4s	AUTORIZADO
P*****5	AUTOMÁTICO	5s	AUTORIZADO
M*****0	AUTOMÁTICO	5s	AUTORIZADO
P*****3	AUTOMÁTICO	4s	AUTORIZADO
I*****9	AUTOMÁTICO	4s	AUTORIZADO
I*****7	AUTOMÁTICO	5s	DESCONHECIDO
D*****8	AUTOMÁTICO	8s	AUTORIZADO
I*****8	AUTOMÁTICO	4s	AUTORIZADO
A*****1	AUTOMÁTICO	8s	DESCONHECIDO
I*****2	MANUAL	10s	DESCONHECIDO
I*****2	MANUAL	10s	AUTORIZADO
I*****1	MANUAL	10s	AUTORIZADO
I*****8	MANUAL	10s	AUTORIZADO
I*****7	MANUAL	10s	DESCONHECIDO
I*****3	MANUAL	10s	AUTORIZADO
I*****2	MANUAL	10s	AUTORIZADO
I*****1	MANUAL	10s	DESCONHECIDO
I*****4	MANUAL	10s	AUTORIZADO
Q*****5	MANUAL	10s	DESCONHECIDO
I*****5	MANUAL	10s	AUTORIZADO

Tabela 3. Tabela de validação do MVP.

Como pode-se aferir, 45% das placas foram identificadas automaticamente com um intervalo médio de, aproximadamente, cinco segundos. Quando o aplicativo demorou mais de dez segundos para fazer a identificação da placa, optou-se por efetuar a identificação manual.

Os motoristas que não foram reconhecidos não estavam no cadastro recebido e, consequentemente, não foram adicionados na base de dados.

Durante os testes, percebeu-se que o aplicativo encontrou dificuldades em diferenciar caracteres com características parecidas, como os caracteres ”1”e ”I”; ”0”e ”O”; ”O”, ”D”e ”Q”, por exemplo. Isso acontece pois a implementação do OpenALPR em Android[13] é baseada em uma versão antiga da biblioteca, onde ainda não havia sido adicionada a compatibilidade com as placas brasileiras. A atualização dessa dependência já está em andamento, mas, de acordo com o criador e mantenedor, ela ainda não pode ser utilizada devido a um problema de gerenciamento de memória[14].

Devido a simplicidade do aplicativo, o teste no estacionamento foi suficiente para aferir sua usabilidade.

[13]OpenALPR Android: https://github.com/SandroMachado/openalpr-android/

[14]Link para descrição do problema: https://github.com/SandroMachado/openalpr-android/pull/41

7. Próximas etapas e melhorias

O escopo do MVP é extremamente restrito. Logo, diversas melhorias podem ser efetuadas nas próximas iterações do projeto. Algumas delas são:

- Implantação nos servidores do IFRS;
- Criação de tutorial para implantação;
- Adição de aplicativo na *Play Store*[15];
- Atualização da biblioteca do OpenALPR para melhorar a identificação de placas brasileiras bem como as placas de duas linhas (motos);
- Integração com o LDAP para permitir que os alunos e servidores possam gerenciar seus próprios veículos;
- Inclusão de fotos dos usuários;
- Cadastro dos seguranças e login no aplicativo;
- Utilização de *WebSockets* para atualizar as permissões de acesso dos usuários;
- Implementação da verificação de turno de acesso ao estacionamento;
- Implementação de autenticações OAuth2[16] para aplicativo e painel administrativo.

8. Considerações finais

Mesmo considerando que o aplicativo não consegue identificar todas as placas automaticamente, o MVP cumpriu o proposto, permitindo que os seguranças identifiquem os veículos e motoristas de maneira simples mesmo quando não há conectividade com a internet.

Se no longo prazo, considerando amostras maiores durante um período mais longo de avaliação, a identificação automática se mostrar ineficiente devido à baixa taxa de conversão e a nova versão da biblioteca ainda não estiver disponível, uma alternativa é utilizar serviços de OCR na nuvem como plano de contingência, requisitando-os toda vez que o aplicativo não conseguir identificar a placa utilizando seu próprio algoritmo. Alguns serviços disponíveis são o Amazon Rekognition[17], OpenALPR Cloud API[18] e Google Cloud Vision API[19] - embora sejam serviços pagos, todos contam com um pacote gratuito que limita a quantidade de requisições mensais. Para redução de custos, pode-se criar um *script* que controle o número de requisições e alterne entre os três serviços, aumentando o número de reconhecimentos gratuitos mensais. A contrapartida dessa solução seria a necessidade de conexão de internet estável.

Referências

Andoid Developers (2018a). Fragments. `https://developer.android.com/guide/components/fragments`. Acessado em 2018-07-09.

Andoid Developers (2018b). Introduction to activities. `https://developer.android.com/guide/components/activities/intro-activities`. Acessado em 2018-07-09.

[15]Loja de aplicativos oficial da plataforma Android
[16]Site do protocolo OAuth2: https://oauth.net/2/
[17]https://aws.amazon.com/pt/rekognition/
[18]http://www.openalpr.com/cloud-api.html
[19]https://cloud.google.com/vision/

Chen, D., Luettin, J., and Shearer, K. (2000). A survey of text detection and recognition in images and videos. https://www.researchgate.net/publication/37432860_A_Survey_of_Text_Detection_and_Recognition_in_Images_and_Videos. Acessado em 2018-06-24.

Cheriet, M., Kharma, N., Liu, C., and Suen, C. (2007). Character recognition systems: A guide for students and practioners. https://pdfs.semanticscholar.org/74d6/68256131f379d63a3d484ccff513f5bbb6d3.pdf. Acessado em 2018-06-24.

Corcos, S. (2017). License plate recognition in react native. https://medium.freecodecamp.org/license-plate-recognition-in-react-native-b4f790d3a160. Acessado em 2018-04-25.

Eikvil, L. (1993). Optical character recognition. https://www.nr.no/~eikvil/OCR.pdf. Acessado em 2018-04-23.

Zareba, B. (2016). How to prepare training files for tesseract ocr and improve characters recognition? http://pretius.com/how-to-prepare-training-files-for-tesseract-ocr-and-improve-character
Acessado em 2018-04-29.