

Aplicação de *Automatic Number Plate Recognition* (ANPR) no controle de acesso de veículos

Maurício de Abreu Cordeiro¹, Alexandro Santos Silva¹

¹Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Avenida Sérgio Vieira Melo, 3150. Bairro Zabelê - Vitória da Conquista - BA - Brasil
CEP 45078-900

mauriciocordeiro@live.com, alexandrossilva@ifba.edu.br

Abstract. *This paper describes the development of an access control system, based on ANPR (Automatic Number Plate Recognition) technologies, with a large potencial of application in physical spaces which demands some security level for access (such as condominium complex and parking lots). The system follows a design based on Microservices Architecture, alongside Docker containers and communication through HTTP following REST principles.*

Resumo. *Este trabalho descreve o desenvolvimento de um sistema para controle de acesso de veículos baseado em ANPR (Automatic Number Plate Recognition), com largo potencial de sua aplicação em locais ou espaços físicos que exigem algum nível de segurança, quando da entrada e saída de pessoas (a título de exemplo, citam-se aqui condomínios e estacionamentos privados). O sistema segue um desenho arquitetural baseado em microsserviços, com contêineres Docker e comunicação via HTTP seguindo os princípios REST.*

1. Introdução

Atualmente, já se tornou comum encontrar ambientes com vídeo-monitoramento, sejam eles interno ou externos. Além disso, os avanços no poder computacional e em técnicas de processamento, detecção e classificação de imagens fez surgir inúmeras possibilidades de inculir semântica e expandir a aplicabilidade de visão computacional em situações do dia a dia.

Nesse cenário, uma das aplicações possíveis é o uso de câmeras para capturar imagens de placas de veículos que passam por guaritas ou portões e, a partir da identificação de sua placa, liberar ou não seu acesso. A tecnologia utilizada para tal é chamada de ANPR (*Automatic Number Plate Recognition*).

Assim, a proposta desse trabalho consiste na implementação de um sistema de controle de acesso com interface *web*, aplicação de ANPR e baseado na arquitetura de microsserviços para a manutenção de uma lista de veículos permitidos, bem como a sua verificação a partir de processamento de imagem.

1.1. Trabalhos correlatos

Em Felix et al. 2017, o ANPR é aplicado em um Circuito Fechado de TV para o monitoramento de entrada e saída de veículos em um ambiente. A solução apresentada utiliza processamento de imagem com MATLAB, OCR (Optical Character Recognition) e rede neural para classificação de imagens e caracteres.

Matysiak et al. 2013 propõem o uso de câmeras com tecnologias de ANPR para monitorar e punir eventuais infrações de trânsito em faixas exclusivas para tráfego de ônibus, na cidade de Varsóvia, na Polônia.

Aalsalem and Khan 2017 apresentam um sistema para monitoramento de estacionamento, com mapeamento e busca de veículos a partir da aplicação de ANPR.

2. Arquitetura

A solução desenvolvida neste trabalho baseia-se na arquitetura de microsserviços, com módulos virtuais containerizados, que se comunicam via HTTP, além da aplicação de tecnologias baseadas em ANPR. Esta seção se dedica a detalhar esses conceitos.

2.1. Arquitetura de Microsserviços

Arquitetura de Microsserviços (*Microservices Architecture* - MSA) é um modelo arquitetural onde processos de *software* são realizados por componentes fracamente acoplados, que possuem funcionalidades específicas e bem definidas, e que se comunicam através de interfaces padronizadas [Viggiato et al. 2018].

A MSA pode ser considerada como a segunda iteração da Arquitetura Orientada a Serviços (*Service Oriented Architecture* - SOA), onde serviços complexos são decompostos em partes mais flexíveis, especializadas e de fácil manutenção, de modo que cada uma é responsável por uma única e pequena funcionalidade com o intuito de realizá-la bem [Hoday et al. 2019].

2.2. Virtualização baseada em contêiner

A Virtualização baseada em contêiner [Eder 2016] consiste na utilização de recursos de *hardware* e do *kernel* de um sistema hospedeiro a fim de criar ambientes isolados para a execução de determinados processos, o chamado **contêiner**, conforme esquematizado na Figura 1.

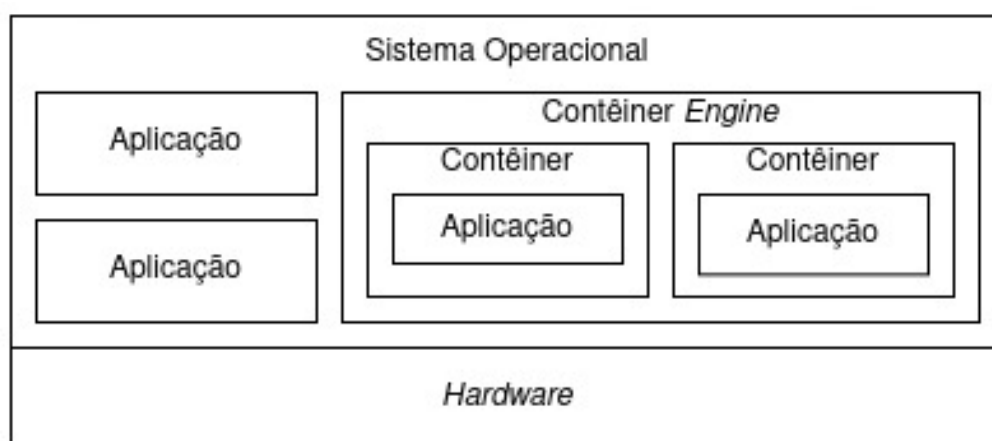


Figure 1. Virtualização baseada em contêiner

Nessa abordagem, o contêiner aloca apenas os recursos necessários para executar sua aplicação, de modo a trabalhar isolado de outros contêineres que possam estar em execução em um mesmo hospedeiro.

2.3. REST

O REST (*REpresentational State Transfer*) é uma implementação da SOA sobre o HTTP para transferência de representações de um determinado recurso entre cliente e servidor [Mumbaikar et al. 2013].

De forma geral, o REST estabelece semântica para métodos e URI (*Universal Resource Identifier*) no HTTP, padronizando a forma como recursos são solicitados e disponibilizados na web (Figura 2). Nesse contexto, leia-se "recurso" como um item acessível via URI, e a "semântica do método" padroniza o modo de interação com o recurso, podendo se associar, por exemplo, POST à criação, GET à solicitação, PUT à edição e DELETE à remoção [Adamczyk et al. 2011].

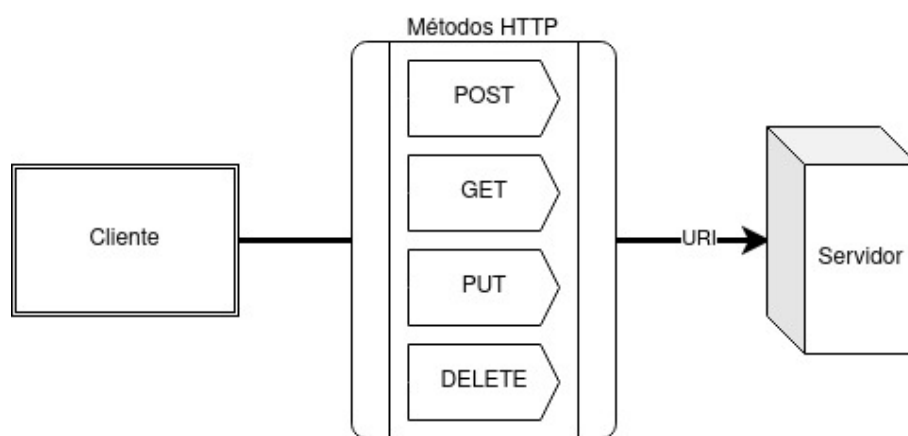


Figure 2. Comunicação cliente-servidor com REST

2.4. ANPR

O ANPR (*Automatic Number Plate Recognition*), que possui papel central na solução desenvolvida neste trabalho, é uma tecnologia que utiliza visão computacional e processamento de imagem para identificar e reconhecer caracteres em placas de veículos.

Existem diferentes técnicas e algoritmos que podem ser usados no ANPR, mas, de forma geral, todos eles seguem um processo com etapas bem definidas [Mufti et al. 2021]: (i) Extração da placa, (ii) Segmentação de caracteres e (iii) Reconhecimento de caracteres.

Em cada uma das etapas para a execução do ANPR, [Shashirangana et al. 2020] pode-se empregar diferentes técnicas (que vão desde algoritmos de processamento de imagens até aplicação de redes neurais para classificação de objetos), conforme o mostrado na Figura 3:

3. Implementação

O sistema para controle de acesso de veículos implementado neste trabalho foi organizado em módulos, ou contêineres, cada um com uma funcionalidade específica (seguindo os princípios da MSA), e se comunicando direta ou indiretamente entre si, como mostrado na Figura 4. Esses módulos foram implementados com o uso de diversas linguagens, *frameworks* e ferramentas, a saber: Docker, Java, Python, Angular e MongoDB.

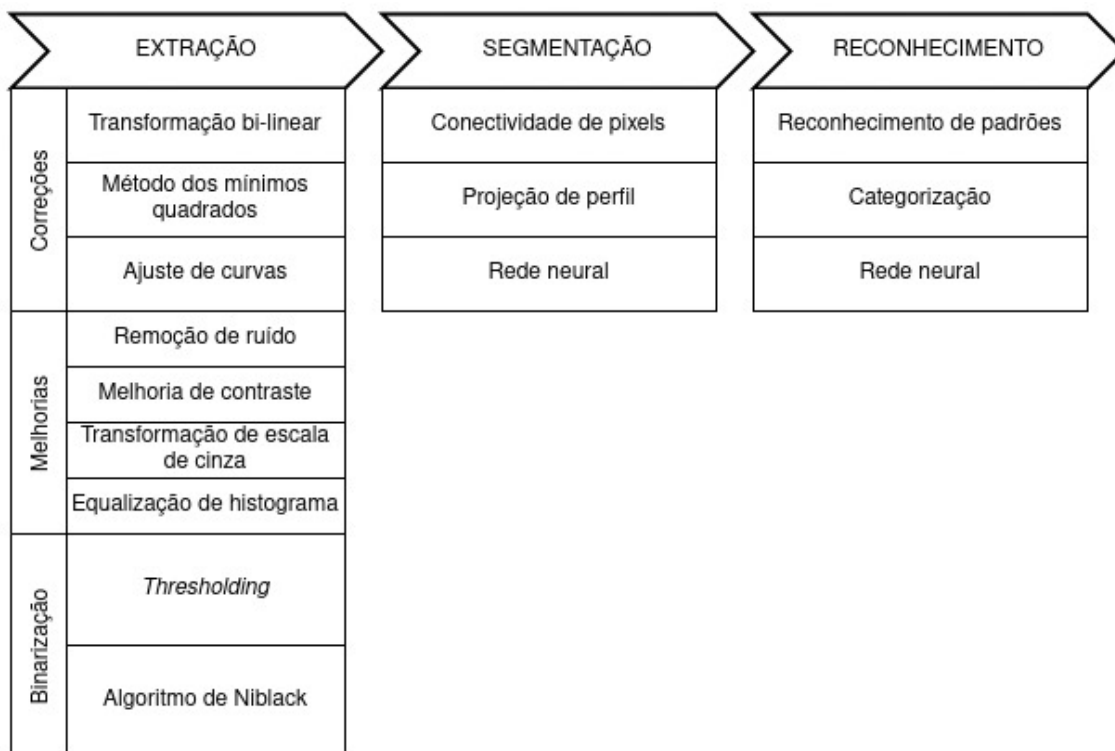


Figure 3. Etapas e técnicas aplicadas no ANPR

A seguir, será apresentada com detalhes a forma com que cada um dos módulos foi desenvolvido e como eles se interagem.

3.1. Composição de contêineres

Todos os módulos do sistema são construídos com contêineres Docker, a partir de imagens padrão ou personalizadas, constituindo assim uma aplicação multi-contêiner.

Para definir e executar a aplicação com múltiplos contêineres, é utilizada a ferramenta Docker Compose¹. Essa ferramenta tem o objetivo de otimizar o processo de construção e execução de múltiplos contêineres, tanto das etapas de desenvolvimento e testes quanto da implantação para produção.

O uso da ferramenta consiste em 3 passos:

1. Definir, no arquivo *Dockerfile*, os recursos necessários de cada um dos contêineres da aplicação, como a imagem docker, dependências, usuários e diretórios, portas de serviços e comandos para execução de aplicações, etc. De forma geral, o *Dockerfile* é um *script* que define **quais** e **como** instalar e utilizar as ferramentas e aplicações de um contêiner;
2. Definir, em um arquivo *.yml*, os serviços necessários e a relação entre os contêineres, como interfaces de rede, portas de serviços, volumes de disco e dependências entre contêineres;
3. Executar o comando `docker-compose up` para inicializar todos os contêineres descritos no arquivo *.yml*

¹Página para a documentação oficial do Docker Compose: <https://docs.docker.com/compose/>

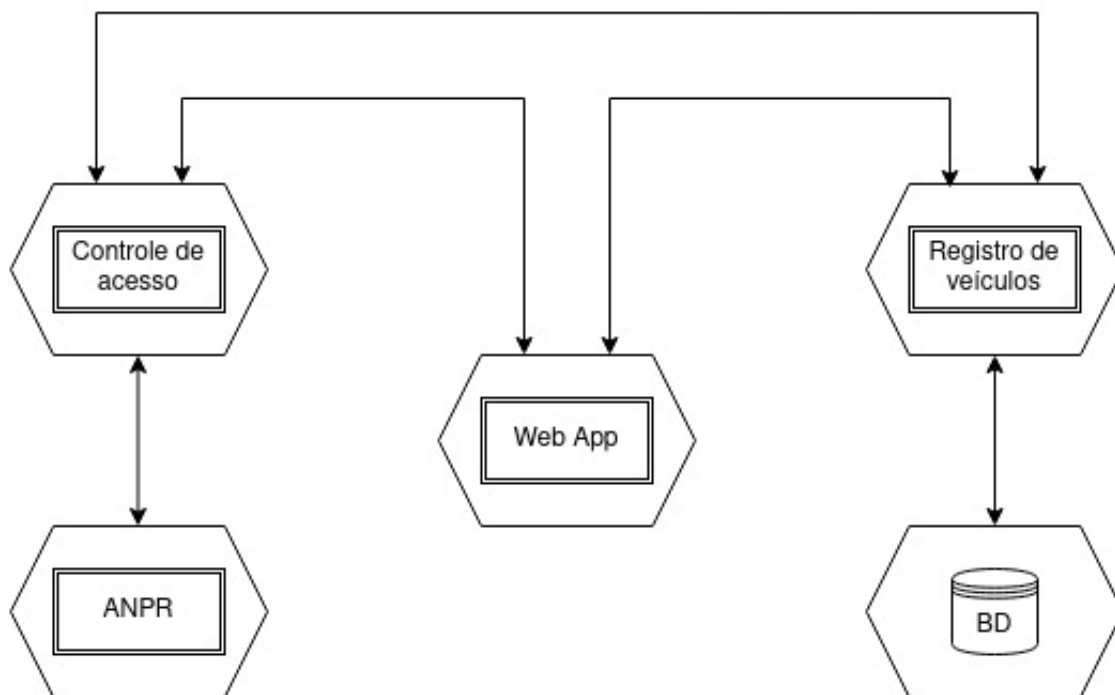


Figure 4. Esquema da organização e relacionamento entre os contêineres do sistema

Vale ressaltar que, apesar do controle ser realizado em um único arquivo, o Docker Compose, em um cenário de atualização, apenas reconstruirá o contêiner que possuir alguma modificação.

3.2. ANPR

O módulo ANPR é o responsável por realizar o processamento das imagens dos veículos e devolver o texto de sua placa. Para tal, foi utilizada a biblioteca de código-aberto OpenALPR² e foi implementada uma API REST com o *framework* Java Spring Boot para processar as requisições de leitura de placas e devolver respostas, no formato JSON³, para o contêiner cliente. A Figura 5 ilustra o fluxo das requisições de leitura de placa para a API do contêiner de ANPR.

A biblioteca OpenALPR, ao receber uma imagem, realiza uma série de etapas para detectar o possível texto escrito na placa do veículo [Hill 2014]:

1. **Deteção:** busca de regiões com possíveis placas na imagem;
2. **Binarização:** transformação em preto-e-branco as regiões de possíveis placas;
3. **Análise de caracteres:** busca por possíveis caracteres na imagem;
4. **Análise de bordas:** busca pelas bordas da placa na região;
5. **Ajuste de distorção:** corrige a perspectiva da placa, para que seja "vista de frente";
6. **Segmentação:** separação dos caracteres da placa em imagens individuais;

²Endereço do repositório e documentação da biblioteca: <https://github.com/openalpr/openalpr>

³JavaScript Object Notation.

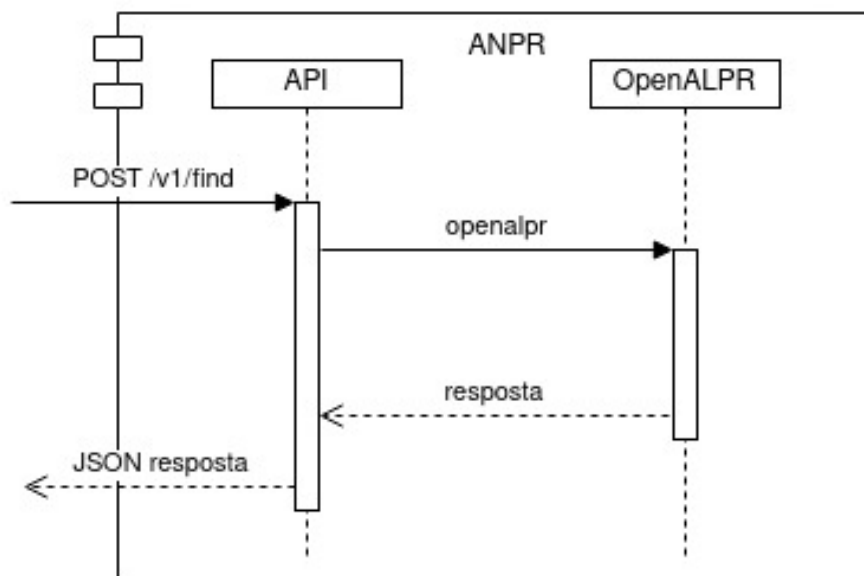


Figure 5. Fluxo de requisições no contêiner de ANPR

7. **OCR:** análise da imagem de cada caractere para encontrar a letra ou número correspondente;
8. **Pós-processamento:** criação de um *ranking* de placas detectadas, com base na confiança e expressões regulares (se solicitado).

A resposta do OpenALPR é então capturada pela API e enviada ao contêiner solicitante.

A adoção do Spring Boot para o desenvolvimento da API deste módulo (bem como do módulo de *Controle de acesso*) se justifica pelo fato do *framework* abstrair uma série de configurações para o desenvolvimento e execução de aplicações Java, enquanto fornece ferramentas para a escrita de funções que utilizam comunicação via HTTP, persistência de dados e suíte de testes, permitindo que o desenvolvedor direcione maior esforço para a implementação de regras de negócio e fluxos semânticos na aplicação.

3.3. Banco de Dados (BD)

Este é o módulo responsável pela persistência dos dados usados pelo sistema, construído com uma imagem do MongoDB⁴. Ele se encontra em um contêiner próprio para fins de controle de eventuais atualizações e uso de volume de disco, uma vez que a lógica da persistência dos dados se encontra no módulo descrito na sub-seção seguinte.

A adoção do MongoDB para a persistência de dados do sistema se deu pelo fato de haver apenas a entidade *Veículo*, sem nenhuma relação com outra entidade dentro do escopo proposto.

3.4. Registro de veículos

Este é o módulo responsável pela lógica de persistência do sistema e acesso ao banco de dados. Escrito em Python, este módulo implementa as operações CRUD⁵ para veículos e

⁴Página oficial do MongoDB: <https://www.mongodb.com/>. Página para a imagem Docker do MongoDB: https://hub.docker.com/_/mongo

⁵Create, Read, Update e Delete

usuários do sistema — utilizando a biblioteca PyMongo⁶ —, bem como implementa uma API — com a biblioteca Flask⁷ —, para tratar as requisições de outros módulos.

O PyMongo é o *driver* oficial do MongoDB para Python, fato que justifica sua escolha. Por sua vez, a adoção do Flask se justifica por ser um *framework* de código-aberto e pela sua versatilidade e independência de outras bibliotecas, mantendo um núcleo simples, porém, extensível.

3.5. Controle de acesso

O módulo de Controle de acesso — escrito em Java com Spring Boot — é onde a principal lógica de negócio do sistema está implementada. Ele é responsável por receber, via requisição REST, uma imagem de uma aplicação cliente qualquer, repassar esta imagem para o módulo ANPR para obter a placa do veículo e então requisitar ao módulo de Registros as informações de cadastro referentes à placa encontrada. De posse dessas informações, ele responde a aplicação cliente informando se o veículo da imagem tem ou não a autorização de acesso. O diagrama da Figura 6 ilustra a troca de mensagens encabeçada pelo módulo de Controle de acesso.

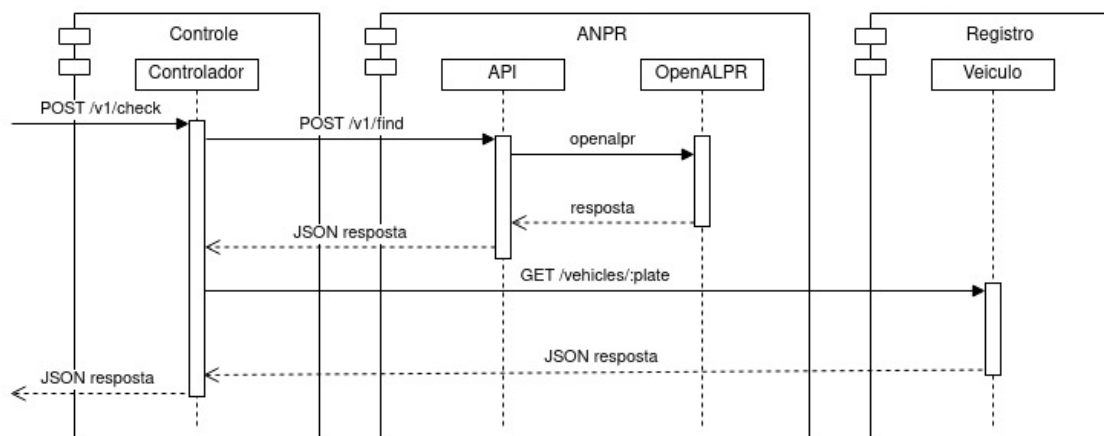


Figure 6. Fluxo de requisições a partir do Controle de acesso

3.6. Web App

O módulo *Web App*, escrito em TypeScript com o *framework* Angular 9, interpreta dois papéis no sistema implementado: (i) *front-end* para a manutenção das informações de cadastro de veículos, acessando o módulo Registro de veículos; e (ii) *front-end* para prova de conceito do sistema, realizando requisições para o módulo de Controle de acesso.

Além de poder ser executado em um contêiner Docker, este módulo pode ser compilado como um aplicativo móvel multiplataforma⁸ através do *framework* Apache Cordova⁹.

⁶Documentação oficial do PyMongo: <https://docs.mongodb.com/drivers/pymongo/>

⁷Documentação oficial do Flask: <https://flask.palletsprojects.com/en/2.0.x/>

⁸Android ou iOS

⁹Documentação oficial do Apache Cordova: <https://cordova.apache.org/docs/en/latest/>

A escolha do Angular para o desenvolvimento deste modo se deu pelo fato de possibilitar a construção de aplicações multi-plataforma de alto desempenho e por possuir uma grande comunidade ativa na Internet.

4. Resultados

O principal objetivo deste trabalho foi implementar um sistema que utilize tecnologias de ANPR baseada em uma arquitetura MSA e que utilize ferramentas amplamente adotadas pela comunidade de desenvolvimento de *software*. Os resultados obtidos são apresentados como se segue.

A construção da infraestrutura interna e a definição das portas de comunicação entre contêineres é realizada através do *Docker Compose* e de seu arquivo de configuração, conforme apresentado anteriormente. Uma vez executado, os 5 contêineres que representam o sistema devem estar acessíveis conforme mostrado na Tabela 1.

	Contêiner	Porta
Banco de dados	db	27017
ANPR	alpr4j	8080
Registro de veículos	vehiclespy	5051
Controle de acesso	check4j	8081
Web App	anpr-ng	4000

Table 1. Contêineres e portas de comunicação

Com o objetivo de demonstrar o conceito do sistema, o módulo *Web Apps* possui três formulários principais, conforme mostrado nas Figuras 7, 8 e 9.

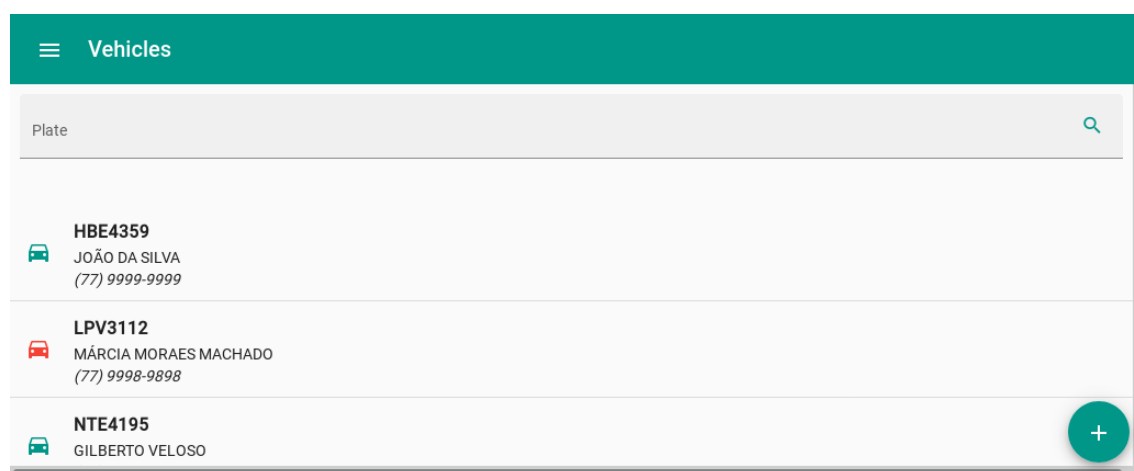


Figure 7. Formulário de listagem de veículos.

Durante o desenvolvimento do módulo ANPR, foi utilizado o *dataset* de Mendes Júnior et al. 2011 para treinamento do OpenALPR. Contudo, devido às características de posicionamento de câmera, iluminação e tipos de placa (placas de uma mesma região e pouca variação de caracteres), a rede neural de detecção não atingiu um resultado satisfatório quando exposta à imagens de um contexto diferente daquele treinado (Tabela 2).

Figure 8. Formulário de cadastro de veículos.

Figure 9. Formulário de verificação de autorização.

Uma vez que o treinamento e aperfeiçoamento dos resultados do módulo ANPR não faz parte do escopo deste trabalho, melhorias no OpenALPR ou até mesmo sua substituição (facilitada pela arquitetura do sistema) serão sugeridas com trabalhos futuros.

5. Trabalhos Futuros

O sistema implementado neste trabalho, apesar de ter cumprido os objetivos propostos, possui alguns pontos de melhoria que podem ser tratados em trabalhos futuros, como:

- **Integração com periféricos de entrada e saída**, a exemplo de câmeras de monitoramento para captura de imagem e vídeo de veículos e com catracas eletrônicas para liberação de veículos;

	Total		Confiança > 90%	
Detectados	467	41,29%	152	32,55%
Não-detectados	664	58,71%	-	-

Table 2. Resultado do teste de validação do OpenALPR, com porcentagem de confiança no resultado.

- **Otimização de gestão e escalabilidade**, pela utilização de ferramentas como Kubernetes¹⁰, para melhor gerenciamento e escalabilidade dos contêineres;
- **Melhoria de performance no módulo de ANPR**: Estudos podem ser realizados neste ponto, com a substituição do contêiner `alpr4j` por outro com uma metodologia diferente de ANPR, com uso de redes neurais que passam por treinamento de acordo com o ambiente onde o sistema será implantado, a fim de melhorar a precisão dos resultados fornecidos pelo módulo;
- **IaC e PaaS**: Implementar uma solução de IaC (*Infrastructure as Code*), como o Terraform¹¹, para gerenciar a construção e implantação do sistema em nuvem (PaaS - Platform as a Service).

References

- Aalsalem, M. Y. and Khan, W. Z. ((2017)). Campussense—a smart vehicle parking monitoring and management system using anpr cameras and android phones. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 809–815. IEEE.
- Adamczyk, P., Smith, P. H., Johnson, R. E., and Hafiz, M. (2011). Rest and web services: In theory and in practice. In *REST: from research to practice*, pages 35–57. Springer.
- Eder, M. (2016). Hypervisor-vs. container-based virtualization. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 1.
- Felix, A. Y., Jesudoss, A., and Mayan, J. A. (2017). Entry and exit monitoring using license plate recognition. In *2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pages 227–231. IEEE.
- Hill, M. (2014). *OpenALPR Design*. OpenALPR, <https://github.com/openalpr/openalpr/wiki/OpenALPR-Design>.
- Homay, A., Zoitl, A., Sousa, M. d., and Wollschlaeger, M. (2019). A survey: Microservices architecture in advanced manufacturing systems. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 1165–1168. IEEE.
- Matysiak, A., Kruszewski, M., Niezgoda, M., and Kamiński, T. (2013). The analysis of anpr camera location points in bus lanes monitoring system in the city of warsaw. *Journal of KONES*, 20.
- Mendes Júnior, P. R., Neves, J. M. R., Tavares, A. I., and Menotti, D. (2011). Towards an automatic vehicle access control system: License plate location. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2916–2921, Anchorage, AK, USA.
- Mufti, N., Shah, S. A. A., et al. (2021). Automatic number plate recognition: A detailed survey of relevant algorithms. *Sensors*, 21(9):3028.
- Mumbaikar, S., Padiya, P., et al. (2013). Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4.

¹⁰Documentação oficial do Kubernetes <https://kubernetes.io/docs/home/>

¹¹Documentação oficial do Terraform: <https://www.terraform.io/docs/index.html>

- Shashirangana, J., Padmasiri, H., Meedeniya, D., and Perera, C. (2020). Automated license plate recognition: a survey on methods and techniques. *IEEE Access*, 9:11203–11225.
- Viggiato, M., Terra, R., Rocha, H., Valente, M. T., and Figueiredo, E. (2018). Microservices in practice: A survey study. *arXiv preprint arXiv:1808.04836*.