RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 14 Nro. 1 2024
Uni-FACEF Centro Universitário Municipal de Franca

Aplicativo Para Visualização e Negociação de Vagas de Estacionamento de Condomínio Residencial com Visão Computacional

Amanda Aparecida de Souza Barboza
Graduanda em Engenharia de Software – Uni-FACEF
amandab.frc@gmail.com

Marcio Maestrelo Funes
Mestre em Computação – Uni-FACEF
marciofunes@facef.br

Resumo

Este artigo apresenta o desenvolvimento de um aplicativo para otimização do uso de vagas de estacionamento em condomínios residenciais. O objetivo é resolver problemas comuns em condomínios com vagas mistas e determinadas, onde a falta de comunicação eficiente entre moradores dificulta a gestão das vagas. Utilizando tecnologias de visão computacional, o aplicativo permite a visualização e negociação de vagas disponíveis. Os métodos incluem a integração de ferramentas de visão computacional para monitoramento e gestão das vagas. Os resultados demonstram a viabilidade de uma aplicação funcional que melhora a gestão de estacionamento, oferecendo uma solução diferenciada para os moradores. Conclui-se que, apesar do potencial promissor, são necessárias otimizações para garantir a escalabilidade e eficiência em ambientes de produção.

Palavras-chave: Aplicativo de estacionamento de condomínio. Negociação de vagas de estacionamento. Visão computacional em estacionamento. Condomínios residenciais inteligentes.

Abstract

This article presents the development of an application to optimize the use of parking spaces in residential condominiums. The objective is to solve common problems in condominiums with mixed and fixed parking spaces, where the lack of efficient communication between residents complicates the management of these spaces. Using computer vision technologies, the application allows the visualization and negotiation of available parking spots. The methods include the integration of computer vision tools for monitoring and managing the spaces. The results demonstrate the feasibility of a functional application that improves parking management, offering a differentiated solution for residents. It is concluded that, despite the promising potential, optimizations are necessary to ensure scalability and efficiency in production environments.

Keywords: *Condominium parking application. Parking space negotiation. Computer vision in parking. Smart residential condominiums.*

1. Introdução

Nos últimos anos, a transformação digital tem impactado significativamente diversos setores da sociedade, incluindo a gestão de condomínios

residenciais. O aumento da urbanização e o crescimento das cidades têm tornado a administração de condomínios uma tarefa cada vez mais complexa, envolvendo desde a manutenção das áreas comuns até a gestão de vagas de estacionamento, um dos principais desafios enfrentados por síndicos e moradores. A falta de um sistema eficiente de controle das vagas pode causar transtornos, especialmente em condomínios com alta rotatividade de veículos ou onde a comunicação entre os residentes é limitada.

Neste contexto, a tecnologia surge como uma solução essencial para melhorar a gestão de vagas de estacionamento. Condomínios com vagas de uso indeterminado enfrentam o desafio de identificar quais estão disponíveis, uma vez que as vagas são ocupadas de forma aleatória. Por outro lado, em condomínios com vagas determinadas, muitas vezes, elas não são plenamente utilizadas, criando oportunidades para que os moradores possam negociar o uso de vagas livres entre si. Ambas as situações evidenciam a necessidade de uma solução tecnológica que facilite a gestão e a comunicação entre os residentes, otimizando o uso desses espaços.

O aplicativo surge como uma solução inovadora para essas questões. Seu objetivo é melhorar a experiência dos moradores e síndicos no gerenciamento das vagas de estacionamento, oferecendo funcionalidades de visualização em tempo real e negociação de vagas. Utilizando tecnologias de visão computacional para monitorar a ocupação das vagas, o aplicativo permite que os usuários visualizem a disponibilidade de vagas, reservem espaços e negociem com outros moradores de maneira prática e transparente.

Este trabalho, portanto, se propõe a desenvolver um aplicativo que atenda a essa demanda crescente por eficiência na gestão de vagas de estacionamento em condomínios. Para isso, o estudo utilizará metodologias que envolvem a integração de ferramentas de visão computacional e banco de dados para monitoramento e gestão das vagas. Os resultados esperados incluem a validação da viabilidade técnica do aplicativo em melhorar a gestão de estacionamento, contribuindo com uma solução diferenciada que traz benefícios tanto para os moradores quanto para a administração do condomínio.

2. Referencial Teórico

Antes de nos aprofundarmos no desenvolvimento do aplicativo, é essencial entender claramente os diferentes aspectos e complexidades associados a esta aplicação. Esta seção tem como objetivo estabelecer um alicerce teórico sólido, abordando tópicos principais sobre o que é visão computacional e como funcionam as vagas de estacionamento em condomínios residenciais. Esta base teórica servirá como um ponto de partida para os materiais e métodos que serão explorados nesta solução.

2.1. Visão Computacional

A Visão Computacional é um campo da ciência da computação que visa automatizar a interpretação e análise de informações visuais, utilizando técnicas que

simulam a capacidade humana de percepção e compreensão. Conforme destacado por Ferrugem (2003), a visão computacional tem como principal objetivo a extração automática de informações relevantes a partir de imagens digitais, permitindo que máquinas e sistemas realizem tarefas como indexação de imagens, reconhecimento de padrões e análise de conteúdo visual.

De acordo com o autor, a indexação automatizada de imagens é uma das áreas centrais da visão computacional, sendo fundamental para o desenvolvimento de sistemas capazes de organizar e recuperar grandes volumes de dados visuais com eficiência. A ideia central é que, a partir de algoritmos de processamento de imagens, seja possível criar descritores que representem as características visuais de uma imagem, facilitando sua identificação e classificação em bancos de dados.

Ferrugem (2003) aponta que os principais desafios da visão computacional envolve questões como variações de iluminação, a presença de ruídos nas imagens, além da necessidade de algoritmos robustos para lidar com diferentes formas, texturas e cores. Esses fatores podem impactar diretamente a precisão dos sistemas, exigindo soluções que combinem técnicas de pré-processamento, segmentação e aprendizado de máquina para melhorar a eficiência e a confiabilidade dos resultados.

Nos últimos anos, o avanço de tecnologias como as redes neurais convolucionais (CNNs) tem impulsionado significativamente o campo da visão computacional, tornando possível a extração de características visuais complexas com uma precisão sem precedentes. Ferrugem (2003) já destacava a importância da automatização na indexação de imagens, um campo que continua evoluindo com o uso de novas abordagens de aprendizado profundo.

Portanto, a visão computacional se posiciona como uma área crítica para o desenvolvimento de aplicações que requerem interpretação e análise de imagens de forma automatizada, sendo essencial tanto para áreas como a robótica, quanto para aplicações na medicina, segurança e automação industrial.

2.2. Estacionamento em Vagas de Condomínio Residenciais

As leis que regulam o uso de vagas de garagem em condomínios são importantes para garantir uma convivência harmoniosa e evitar conflitos entre os moradores. A Lei Federal 12.607/2012, que alterou o Código Civil Brasileiro, desempenha um papel fundamental nesse contexto ao estabelecer que as vagas de garagem podem ser classificadas como áreas comuns ou privativas, dependendo de sua natureza e destinação.

As vagas determinadas são aquelas com matrícula individualizada no cartório de imóveis, ou seja, fazem parte do patrimônio do proprietário da unidade condominial e podem ser livremente utilizadas, vendidas ou alugadas, desde que respeitem as regras estabelecidas pela convenção do condomínio. Esse tipo de vaga está diretamente vinculado à propriedade, garantindo maior autonomia ao condômino sobre sua gestão.

As vagas mistas, por outro lado, não possuem matrícula própria e são de uso comum. Elas são distribuídas entre os condôminos de acordo com critérios definidos pela convenção ou assembleia, como sorteio ou rodízio. Por serem de uso coletivo, a venda ou aluguel dessas vagas é vedada, e sua administração depende da gestão condominial.

Já as vagas autônomas, que também possuem matrícula própria, não estão necessariamente vinculadas a uma unidade residencial. Elas podem ser negociadas de forma independente, permitindo ao proprietário vender ou alugar a vaga até mesmo para terceiros que não residem no condomínio, desde que respeitem as normas condominiais e obtenham aprovação prévia, se exigida pela convenção.

Além da Lei 12.607/2012, a Constituição Federal e o Código Civil também influenciam a regulamentação das vagas de garagem ao definirem os direitos de propriedade, o uso das áreas comuns, e o papel das convenções condominiais na resolução de conflitos e na gestão das áreas de uso coletivo.

3. Materiais e métodos

Nesta seção, será abordado em detalhes a criação dos materiais e os métodos necessários para o desenvolvimento do aplicativo, sendo descritos o diagrama de entidade relacionamento e as tecnologias utilizadas.

3.1. Diagrama Entidade Relacionamento

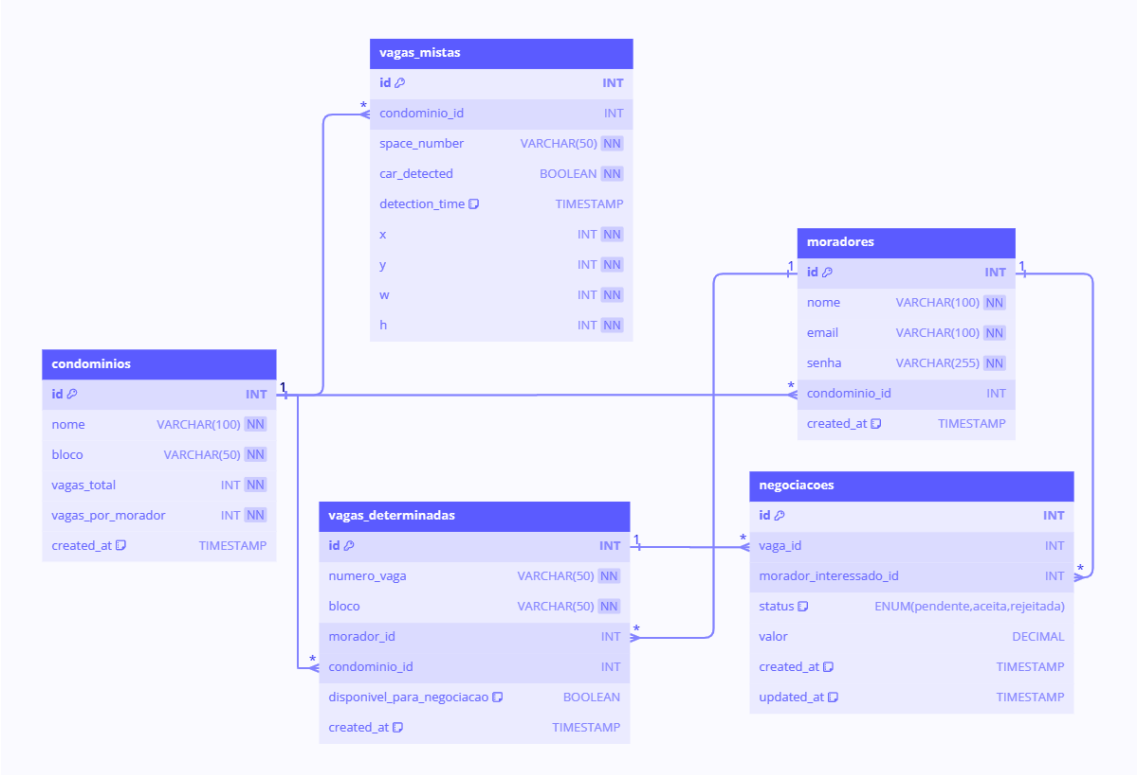
O Diagrama Entidade-Relacionamento (DER) apresentado a seguir, Figura 1, é fundamental para a estruturação do aplicativo, sendo criado o banco de dados a partir dele. Este modelo reflete a inter-relação entre as diferentes entidades que compõem o sistema e suas respectivas propriedades, proporcionando uma visão clara das funcionalidades e interações dos usuários.

Entidades e Relacionamentos

1. **Condomínios**: A entidade **condominios** representa os condomínios que participam do sistema. Cada condomínio possui um identificador único **(id)**, um nome **(nome)**, um bloco **(bloco)**, e a quantidade total de vagas disponíveis **(vagas_total)**. Além disso, especifica o número de vagas que cada morador pode ocupar **(vagas_por_morador)**, permitindo uma gestão eficiente do espaço disponível.]

2. **Moradores**: A entidade **moradores** armazena informações sobre os residentes de cada condomínio. Cada morador é identificado por um **id**, possui um nome **(nome)**, um e-mail **(email)**, e uma senha **(senha)** para acesso ao sistema. O relacionamento com a entidade **condominios** é estabelecido através do **condominio_id**, que indica a qual condomínio o morador pertence.

Figura 1 - Diagrama Entidade Relacionamento do Banco de Dados



Fonte: A Autora

3. **Vagas Determinadas**: A entidade **vagas_determinadas** relaciona as vagas de estacionamento atribuídas a cada morador. Cada vaga é identificada por um **id** e possui informações como o número da vaga **(numero_vaga),** o bloco ao qual pertence **(bloco)**, e o **morador_id** que a ocupa. Também, há um campo **(disponivel_para_negociacao)** que indica se a vaga está disponível para negociação, permitindo que os moradores tenham flexibilidade na gestão de suas vagas.

4. **Vagas Determinadas**: A entidade **vagas_determinadas** relaciona as vagas de estacionamento atribuídas a cada morador. Cada vaga é identificada por um **id** e possui informações como o número da vaga (**numero_vaga**), o bloco ao qual pertence (**bloco**), e o **morador_id** que a ocupa. Além disso, há o campo **disponivel_para_negociacao** que indica se a vaga está disponível para negociação, permitindo que os moradores tenham flexibilidade na gestão de suas vagas.

5. **Negociações**: A entidade **negociacoes** registra os processos de negociação de vagas. Cada negociação é vinculada a uma vaga específica (**vaga_id**) e a um morador interessado (**morador_interessado_id**). O status da negociação pode variar entre "pendente", "aceita" e "rejeitada", permitindo um acompanhamento claro das transações. Também é possível registrar o valor da negociação.

6. **Vagas Mistas**: A entidade **vagas_mistas** é responsável por monitorar o status das vagas em tempo real. Com campos como **space_number**, **car_detected**, e informações sobre a detecção de objetos (coordenadas x, y, largura e altura), esta entidade permite que o sistema tenha uma visão precisa sobre quais vagas estão ocupadas, potencializando a experiência dos moradores na busca por vagas disponíveis.

Essa estrutura robusta é capaz de atender às necessidades de gestão de vagas em condomínios. A inter-relação entre as entidades proporciona um fluxo de informações eficiente, possibilitando que moradores visualizem, negociem e acompanhem o status de suas vagas. A abordagem sistemática não apenas melhora a experiência do usuário, mas também otimiza a utilização dos espaços de estacionamento, contribuindo para um gerenciamento mais eficiente dos recursos disponíveis.

3.2. Tecnologias utilizadas

Nesta seção, foi abordado as tecnologias empregadas no desenvolvimento do aplicativo, detalhando as motivações para a escolha de cada uma, bem como seus benefícios e funcionalidades. O aplicativo foi construído utilizando uma combinação de *React Native* para a interface do usuário e *Python* com a biblioteca *OpenCV* para as funcionalidades de visão computacional. Esses elementos foram integrados para oferecer uma solução eficiente para a gestão de vagas de estacionamento em condomínios residenciais. A seguir, foi apresentado as características e vantagens de cada uma dessas tecnologias.

3.2.1. React Native

Para o desenvolvimento do aplicativo, optou-se pelo uso do *React Native*, uma biblioteca *JavaScript* que possibilita a criação de aplicativos móveis nativos para *Android* e *iOS* a partir de um único código-base. Segundo Norbert Kamienski (2023), essa tecnologia permite um desenvolvimento mais rápido e econômico, mantendo a performance e a flexibilidade de uma aplicação nativa. Utilizando o *React Native*, foi possível desenvolver interfaces responsivas e intuitivas, reutilizando componentes e assegurando uma experiência uniforme para os usuários em ambas as plataformas.

Essa biblioteca conta com uma comunidade ativa e uma vasta quantidade de templates prontos para uso, o que facilita a resolução de problemas e a implementação de novas funcionalidades. Hitesh Agarwal (2023) destaca que essa escolha é popular entre desenvolvedores devido à sua capacidade de criar aplicativos com interfaces de usuário atraentes e desempenho semelhante ao de aplicativos nativos. Além disso, a tecnologia permite a integração com módulos nativos, expandindo suas capacidades e possibilitando o uso de funcionalidades específicas de cada plataforma.

A reutilização de componentes foi essencial para garantir a consistência visual e funcional do aplicativo. A abordagem modular permitiu a criação de componentes reutilizáveis, que foram adaptados conforme necessário para atender às especificidades de cada plataforma. Isso não apenas acelerou o processo de

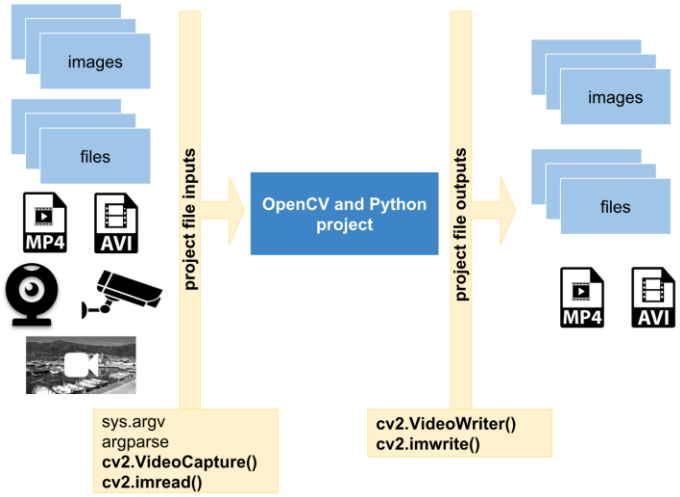
desenvolvimento, mas também assegurou que as atualizações e manutenções futuras fossem mais eficientes.

3.2.2. Python e OpenCV

Python tem se destacado como uma linguagem de programação *back-end* essencial para o desenvolvimento de soluções em visão computacional, especialmente em aplicações práticas como a gestão de estacionamentos de condomínios. Sua sintaxe simples e a rica disponibilidade de bibliotecas facilitam a implementação de algoritmos avançados, cruciais para a análise e o processamento de imagens em sistemas de monitoramento e controle de vagas.

De acordo com Souza (2024), Python é amplamente adotado em projetos de visão computacional devido à sua integração com bibliotecas poderosas como o *OpenCV (Open Source Computer Vision Library)*. O *OpenCV* fornece uma ampla gama de ferramentas para manipulação e análise de imagens, que são essenciais para o desenvolvimento de sistemas capazes de automatizar a detecção de vagas, a identificação de veículos e o reconhecimento de placas em estacionamentos de condomínios, como é possível observar no fluxograma da Figura 2, mostrando como é feita a leitura e processamento de arquivos.

Figura 2 - Fluxograma de Processamento de Arquivos com *OpenCV* e *Python*



Fonte: VILLAN, Alberto Fernandez. Mastering OpenCV 4 with Python. 2019.

O *OpenCV* é amplamente utilizado em sistemas de reconhecimento de imagens, que envolvem a identificação e a classificação de objetos em imagens digitais. Esse processo pode ser dividido em várias etapas importantes. A primeira delas é o pré-processamento, que é realizado antes da aplicação de qualquer algoritmo de reconhecimento. Nessa etapa, as imagens geralmente precisam ser preparadas para melhorar a qualidade dos dados. Isso pode incluir a conversão para escala de cinza, a equalização de histograma e a remoção de ruídos, técnicas que facilitam uma análise mais precisa (GONZALEZ; WOODS, 2018).

Outra etapa essencial é a detecção de características, na qual o *OpenCV* emprega diversos algoritmos para identificar características importantes nas imagens, como bordas, pontos-chave e descritores. Entre as técnicas utilizadas estão a Transformada de Hough e os detectores de bordas de Canny, que permitem a

identificação de características distintas, cruciais para a análise subsequente (SZELISKI, 2021). Segundo Dalal e Triggs (2005), a detecção de bordas e a utilização de descritores como Histogram of Oriented Gradients (HOG) desempenham um papel fundamental na identificação de objetos em sistemas de visão computacional.

Após a detecção de características, o próximo passo envolve a classificação dos objetos. O *OpenCV* suporta diferentes métodos de aprendizado de máquina e redes neurais para o treinamento de modelos que conseguem reconhecer variados tipos de objetos. No contexto de um sistema de estacionamento, isso pode incluir a identificação de diferentes tipos de veículos, o reconhecimento de placas de licença e a verificação da presença de veículos em vagas específicas (LU; PLATEK, 2019).

Por fim, para o monitoramento contínuo, o *OpenCV* oferece técnicas de rastreamento de objetos, permitindo o acompanhamento da posição dos veículos ao longo do tempo. Algoritmos como KLT (Kanade-Lucas-Tomasi) e o Mean Shift podem ser utilizados para seguir o movimento dos veículos, garantindo uma gestão eficiente das vagas (DORNAIKA; EL BAZ, 2020).

A utilização dessas técnicas em *Python* para o gerenciamento de estacionamento de condomínios permite a criação de funcionalidades que não apenas detectam a ocupação das vagas, mas também identificam e monitoram veículos de forma eficaz. Isso pode incluir a representação da utilização das vagas, alertas para vagas ocupadas indevidamente e até mesmo a automação de processos de entrada e saída com base na identificação dos veículos.

A flexibilidade e a compatibilidade do *Python* com diferentes plataformas facilitam a integração dessas soluções em diversas infraestruturas tecnológicas, permitindo que sejam adaptadas a diferentes tamanhos de estacionamentos e necessidades específicas de cada condomínio.

Neste trabalho, *Python* é empregado para implementar soluções de visão computacional destinadas ao gerenciamento de estacionamento em condomínios. Utilizando técnicas descritas por Antonello (2019), um dos objetivos do aplicativo é desenvolver uma funcionalidade para condomínios de vagas mistas que possa detectar e monitorar a ocupação de vagas, bem como identificar e registrar a presença de veículos de forma eficiente e precisa.

3.3. MySQL

O *MySQL* é um sistema de gerenciamento de banco de dados relacional (SGBDR) amplamente utilizado, desenvolvido inicialmente pela empresa sueca MySQL AB e atualmente mantido pela *Oracle Corporation*. Trata-se de um software de código aberto, disponível gratuitamente, reconhecido por sua eficiência, confiabilidade e facilidade de uso (DUBois, 2013). Desde sua aquisição pela *Oracle*, o *MySQL* tem sido continuamente aprimorado, com novos recursos de segurança, desempenho e escalabilidade, o que o torna uma escolha popular para uma ampla gama de aplicações, desde pequenos projetos até sistemas corporativos robustos (VAIDHEESWARAN; GUPTA, 2019).

No desenvolvimento, o *MySQL* foi a escolha ideal para o banco de dados. Como um SGBDR, o *MySQL* oferece alta performance e escalabilidade, essenciais para o crescimento do sistema à medida que mais usuários e condomínios são integrados (KALLAS; THOMPSON, 2018). Em adição, sua arquitetura segura, com suporte a transações e integridade referencial, garante que os dados armazenados, como informações sobre os condomínios, vagas disponíveis, usuários e histórico de negociações, sejam protegidos e gerenciados de maneira eficiente (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

A estrutura das tabelas foi projetada para armazenar informações detalhadas dos condomínios, com campos específicos para cada vaga, como localização, status (disponível/ocupada) e histórico de negociações. As consultas SQL também foram cuidadosamente otimizadas para garantir acesso rápido e eficiente aos dados, especialmente em operações de consulta, inserção e atualização, típicas de um sistema de gestão de vagas. Essas consultas foram desenhadas com o objetivo de minimizar o tempo de resposta e melhorar a experiência do usuário, mesmo em cenários com um grande volume de dados (Date, 2004).

4. Desenvolvimento

Nesta seção, foi explorada a aplicação prática com o desenvolvimento de técnicas de visão computacional com biblioteca *OpenCV* em conjunto com a linguagem *Python* para a seleção e automação da detecção das vagas do estacionamento.

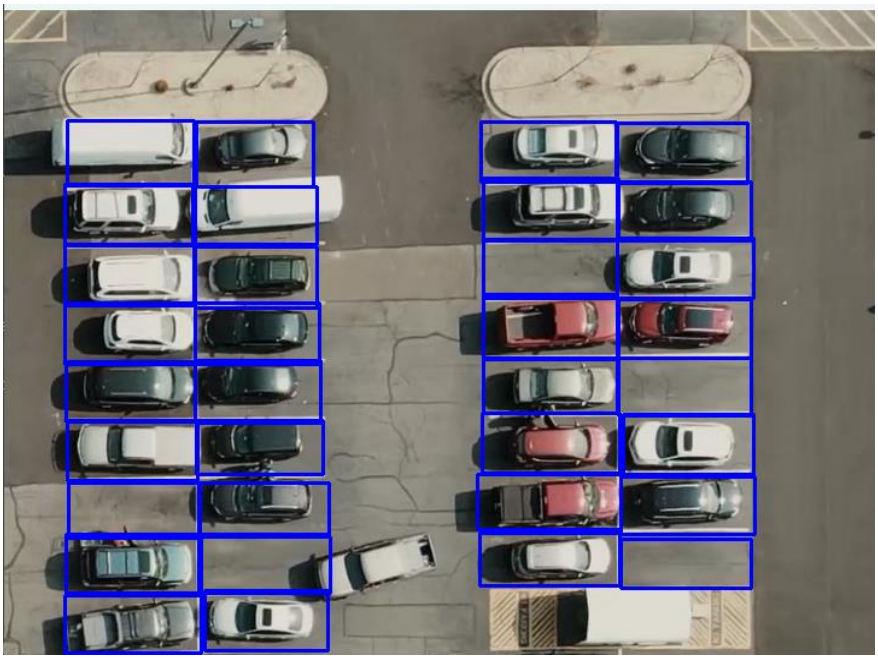
4.1. Demarcação de Vagas

No contexto de vagas mistas, a detecção das vagas do estacionamento requer uma visão aérea, obtida por meio de câmeras instaladas no local. Para que o computador aprenda a identificar as coordenadas que deve monitorar, a fim de determinar se uma vaga está disponível ou ocupada, é necessário processar uma imagem capturada no mesmo ângulo, posição e área do vídeo da câmera que realizará a detecção. Nisso, é realizada a demarcação da área de cada vaga.

Na Figura 3, é o início da utilização do *OpenCV*. Lembrando que essa parte é realizada pelo desenvolvedor e não por usuários externos, pois não existe essa tela no aplicativo, para o usuário final será apresentado como na Figura 16. Primeiro é importada uma imagem do tipo JPG ou PNG, que esteja no mesmo ângulo e local do vídeo da câmera que fará o monitoramento, para a biblioteca processar e apresentar na tela.

Nessa imagem processada, deve conter toda a área do estacionamento e é realizada a demarcação de retângulos azuis em cada área onde o algoritmo deverá identificar a mudança de tons de branco e preto para notar a presença do carro, sendo gravadas as coordenadas no arquivo **vagas.pkl** para depois acessar durante o processo de detecção. Ou seja, se existirem 50 vagas, serão realizadas 50 demarcações. É possível visualizar essa lógica criada no código da Figura 4 e a inserção no na tabela de vagas mistas do banco de dados das vagas selecionadas na Figura 5.

Figura 3 - Visão aérea do estacionamento com demarcação das áreas onde estão as vagas



Fonte: A Autora

Figura 4 - Código da seleção das vagas

```python
# Carregando a imagem
img = cv2.imread('data/estacionamento.png')
vagas = []

# Selecionando as vagas manualmente
for _ in range(50):  # Alterado de x para _ para evitar confusão com o índice
    vaga = cv2.selectROI('vagas', img, False)
    cv2.destroyWindow('vagas')
    vagas.append(vaga)

    for (x, y, w, h) in vagas:  # Certifique-se de que o loop está correto
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Salvando as vagas no arquivo
with open('vagas.pkl', 'wb') as arquivo:
    pickle.dump(vagas, arquivo)

# Verificando conteúdo das vagas antes da inserção
print("Conteúdo de vagas:", vagas)

# Inserindo as vagas no banco de dados
for idx, (x, y, w, h) in enumerate(vagas, start=1):  # start=1 para começar o número das vagas a partir de 1
    query = f'''
    INSERT INTO car_detection (space_number, x, y, w, h, car_detected)
    VALUES ({idx}, {x}, {y}, {w}, {h}, 0)
    ON DUPLICATE KEY UPDATE
        x = VALUES(x),
        y = VALUES(y),
        w = VALUES(w),
        h = VALUES(h),
        car_detected = VALUES(car_detected);
    '''
    print(f"Executando query: {query}")  # Print da query para debug
    execute_query(query)
```

Fonte: A Autora

Figura 5 - Registros na tabela de vagas mistas do banco de dados

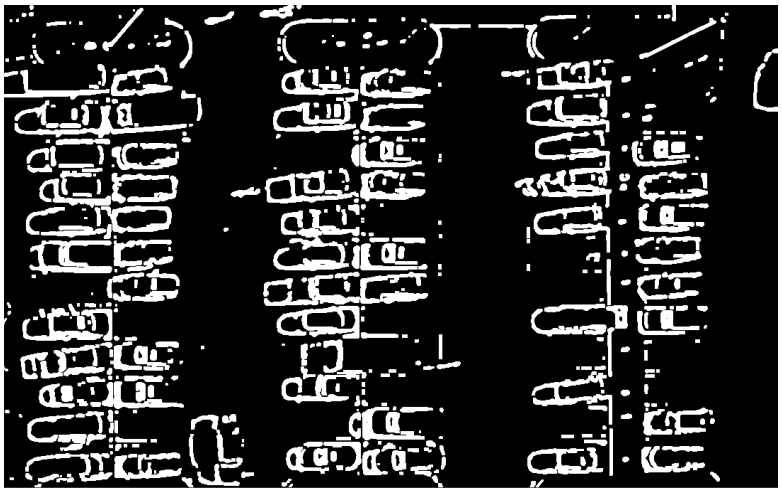
id	space_number	car_detected	detection_time	x	y	w	h
1	0	1	2024-09-08 19:21:39	164	622	101	45
2	1	1	2024-09-08 19:21:39	162	570	104	51
3	2	1	2024-09-08 19:21:39	163	526	108	43
4	3	1	2024-09-08 19:21:39	159	477	112	46
5	4	1	2024-09-08 19:21:39	161	426	113	51
6	5	1	2024-09-08 19:21:39	159	382	115	44
7	6	1	2024-09-08 19:21:39	163	332	110	50
8	7	1	2024-09-08 19:21:39	160	288	112	42
9	8	1	2024-09-08 19:21:39	158	238	116	49
10	9	1	2024-09-08 19:21:39	161	190	114	49
11	10	1	2024-09-08 19:21:39	159	142	120	47

Fonte: A Autora

4.2. Detecção de Vagas

Após a demarcação das vagas na Figura 3, é realizado o processamento de imagem. A Figura 6, apresentada abaixo, é uma representação binária do estacionamento, processada para facilitar a detecção das vagas disponíveis através de técnicas de visão computacional.

Figura 6: Imagem Binária do Estacionamento



Fonte: A Autora

A imagem foi gerada utilizando técnicas de limiarização, onde cada pixel é transformado em preto ou branco com base em um valor de limiar específico - podendo ser identificada no código da Figura 7. Esse processo de binarização é crucial para segmentar a imagem e identificar diferentes elementos, como veículos e vagas de estacionamento. O alto contraste entre o fundo preto e os contornos brancos facilita a detecção de carros estacionados por meio de algoritmos de visão computacional.

Figura 7 - Conversão de cada frame para escala de cinza e processamento para destacar as áreas de interesse

```
imgCinza = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgTh = cv2.adaptiveThreshold(imgCinza, maxValue: 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
                              blockSize: 25, C: 16)
imgMedian = cv2.medianBlur(imgTh, ksize: 5)
kernel = np.ones( shape: (3, 3), np.int8)
imgDil = cv2.dilate(imgMedian, kernel)
```

Fonte: A Autora

Na Figura 7, código dessa detecção, mostra que primeiro a imagem é convertida para escala de cinza **(imgCinza)** para facilitar a manipulação. Depois um limiar adaptativo é aplicado à imagem em cinza para converter os tons de cinza em uma imagem binária, onde as vagas são destacadas **(imgTh)** e um filtro de mediana é utilizado para reduzir o ruído da imagem binária **(imgMedian)**. Por último, a imagem é dilatada para preencher lacunas em áreas de interesse, facilitando a contagem de pixels brancos **(imgDil)**.

A identificação das vagas do estacionamento é realizada através dessa análise da imagem binária. Algoritmos de visão computacional, como a detecção de bordas e contornos, são empregados para localizar áreas livres (em preto) e ocupadas (em branco). Também, técnicas de análise de formas são utilizadas para diferenciar veículos de outros objetos presentes na imagem.

Este código da Figura 8 utiliza *OpenCV* para processar o vídeo (simulando a câmera do condomínio) e detectar a ocupação de vagas de estacionamento previamente demarcadas. Ele carrega as coordenadas das vagas do arquivo **vagas.pkl**, que tem armazenado as coordenadas das vagas que foram selecionadas na Figura 3, e lê o vídeo frame a frame. Cada frame é convertido para escala de cinza e, em seguida, passa por operações de limiar adaptativo, filtragem e dilatação para realçar as áreas das vagas.

Como é possível observar na Figura 9, também é verificada a ocupação das vagas contando os pixels brancos em cada área correspondente a uma vaga. Se a contagem for menor que 900, a vaga é considerada livre e um retângulo verde é desenhado, caso contrário, é considerada ocupada com um retângulo vermelho. Quando o status de uma vaga muda, o código atualiza esse status no banco de dados, executando uma consulta SQL que modifica o campo **car_detected** na tabela **vagas_mistas**.

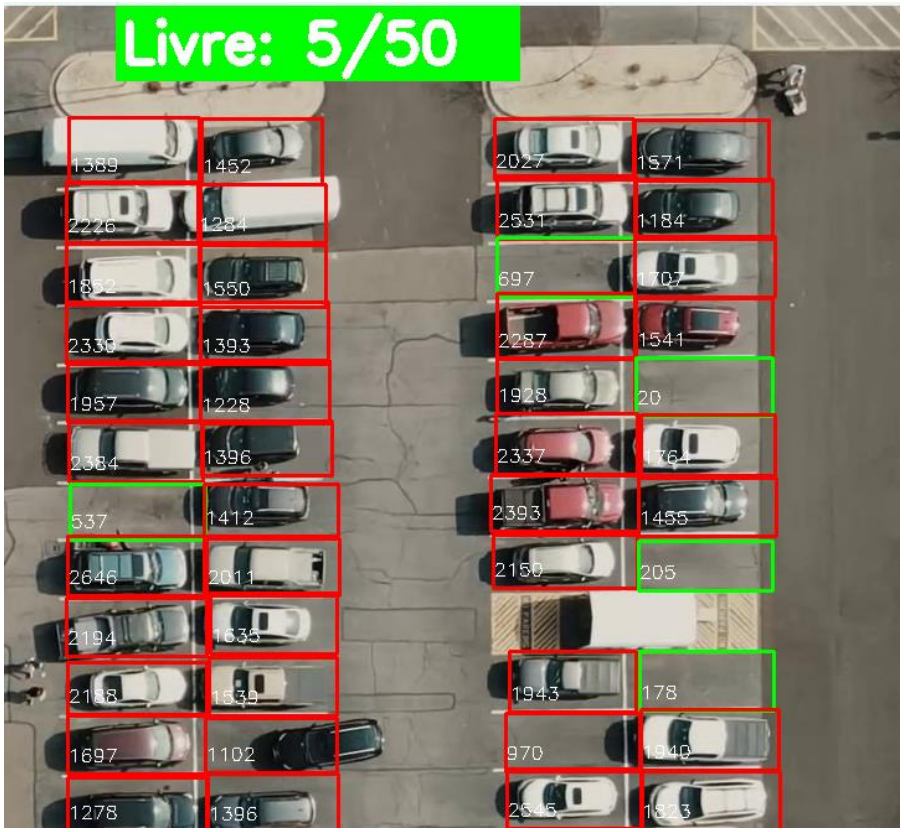
Durante a desenvolvimento foram criados esses retângulos verdes e vermelhos e monitoramento do número total de vagas livres a fim de comparação com a atualização no banco de dados, para testes, pois é de fácil visualização. Para o usuário, é consumida essas informações no *back-end* e a interface apresentada está presente na Figura 16.

Figura 8 - Código que identifica, conta e atualiza as vagas em tempo real no banco de dados

```
vagas = []
with open('vagas.pkl', 'rb') as arquivo:
    vagas = pickle.load(arquivo)

video = cv2.VideoCapture('data/video.mp4')

wait_time = 1

# Armazenando o status das vagas detectadas anteriormente
status_anterior = [None] * len(vagas)

while True:
    check, img = video.read()

    if not check:
        break

    imgCinza = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgTh = cv2.adaptiveThreshold(imgCinza, maxValue: 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
                                  blockSize: 25, C: 16)
    imgMedian = cv2.medianBlur(imgTh, ksize: 5)
    kernel = np.ones( shape: (3, 3), np.int8)
    imgDil = cv2.dilate(imgMedian, kernel)

    vagasAbertas = 0

    for idx, (x, y, w, h) in enumerate(vagas):
        vaga = imgDil[y:y + h, x:x + w]
        count = cv2.countNonZero(vaga)
        cv2.putText(img, str(count), org: (x, y + h - 10), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
                    color: (255, 255, 255), thickness: 1)

        if count < 900:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            vagasAbertas += 1
            status = 0  # Vaga livre
        else:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
            status = 1  # Vaga ocupada

        # Atualiza o status se houve uma mudança
        if status_anterior[idx] is None or status_anterior[idx] != status:
            # Atualizando o status da vaga no banco de dados
            query = f'''
            UPDATE vagas_mistas
            SET car_detected = {status}
            WHERE space_number = {idx};
            '''
            execute_query(query)
            status_anterior[idx] = status

    # Mostrando o total de vagas livres
    cv2.rectangle(img, (90, 0), (415, 60), (0, 255, 0), -1)
    cv2.putText(img, text: f'Livre: {vagasAbertas}/{len(vagas)}', org: (95, 45), cv2.FONT_HERSHEY_SIMPLEX,
```

Fonte: A Autora

Figura 9 - Detecção das vagas livres e ocupadas



Fonte: A Autora

5. Resultados

Nesta seção, serão apresentados os resultados das telas desenvolvidas com as funcionalidades do *back-end* já integradas ao aplicativo, trazendo os dados vindos do banco de dados.

5.1. Telas do Aplicativo

Para compreender melhor as funcionalidades e a interface de um aplicativo móvel voltado para a gestão de estacionamento de condomínios, é importante analisar as telas que compõem o processo de cadastro e configuração inicial. A imagem da Figura 10 ilustra a primeira tela do aplicativo, a tela de *login*.

Figura 10 - Tela de *Login*

Fonte: A Autora

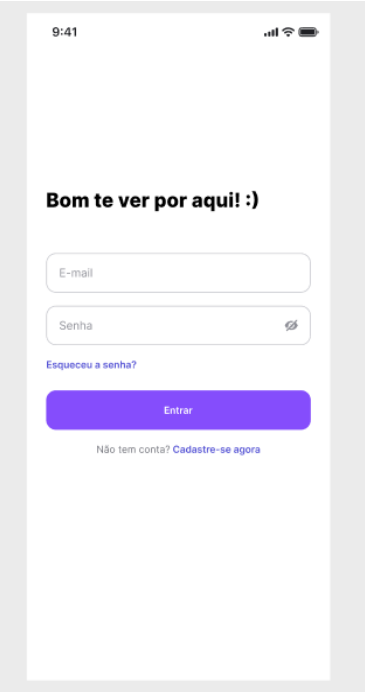
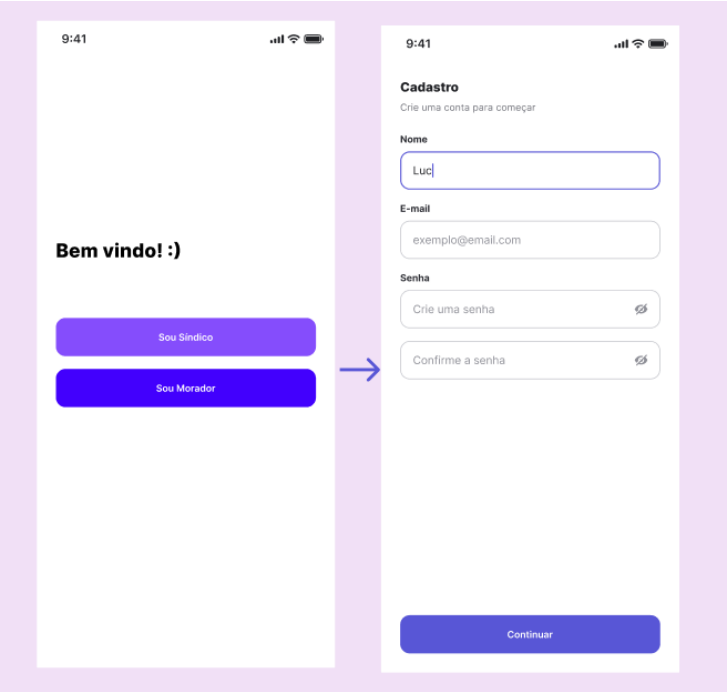


Figura 11 - Telas de Cadastro do Usuário

Fonte: A Autora



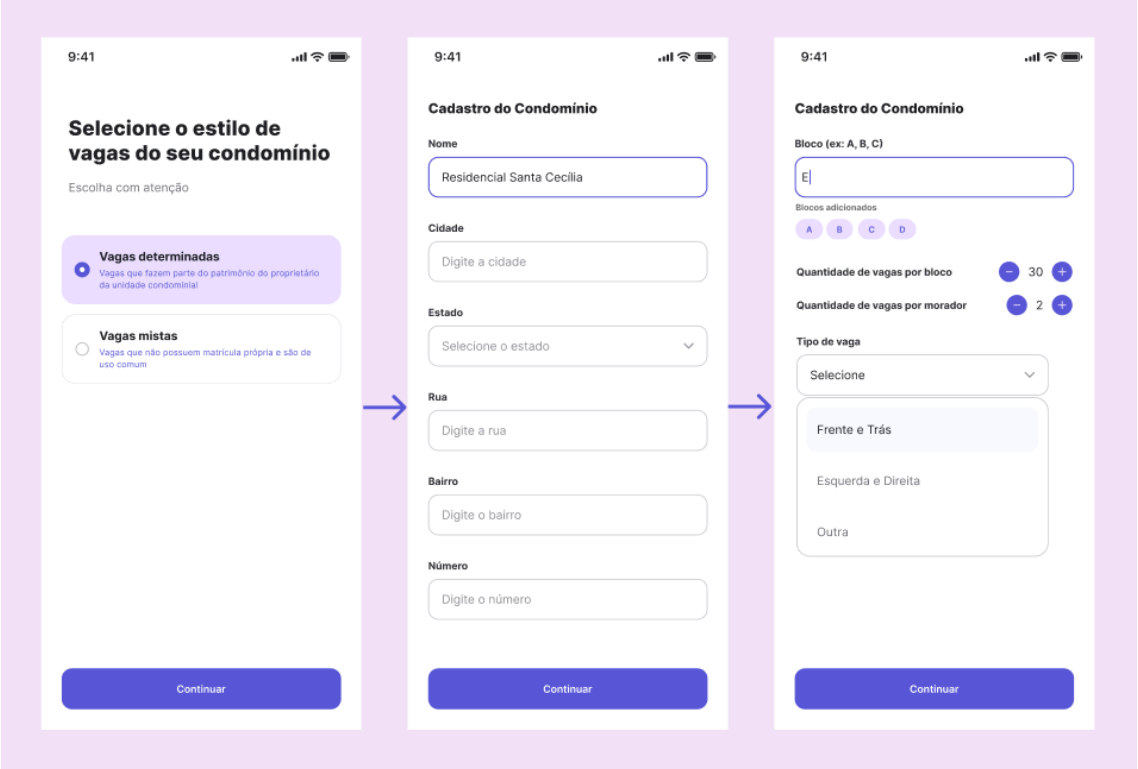
A Figura 10 apresenta a tela de *login* do usuário. Nesta tela, o usuário deve preencher seu e-mail e senha previamente cadastrados para acessar o aplicativo. Caso ainda não possua uma conta, ele pode clicar no botão “Cadastre-se agora”, que o redireciona para a primeira tela da Figura 11 para iniciar o cadastro.

Na primeira tela, à esquerda, da Figura 11 o usuário é recebido com a mensagem “Bem vindo!” e pode escolher entre os botões “Sou Síndico” e “Sou Morador”, selecionando seu papel no condomínio - ambas mandam para a tela de cadastro de usuário, porém apenas o síndico terá acesso a tela de cadastro do condomínio presente na Figura 12.

Na segunda tela, há um formulário de cadastro de usuário, sendo a mesma tela para síndicos e moradores, onde é necessário preencher nome, e-mail, senha e confirmação de senha. Um botão “Continuar” está localizado na parte inferior da tela para prosseguir com o cadastro. Se a pessoa que estiver se cadastrando for morador, será redirecionado para a tela da Figura 13.

Depois do síndico fazer o cadastro de usuário, é redirecionado para a primeira tela da Figura 12, que inicia o fluxo do cadastro do condomínio, no qual primeiro ele deve selecionar o tipo de modalidade correspondente das vagas do condomínio. A segunda tela exibe um formulário com os campos de endereço do condomínio, solicitando informações como nome do condomínio, endereço completo (rua, bairro, número, cidade e estado).

Figura 12 - Telas de Cadastro do Condomínio



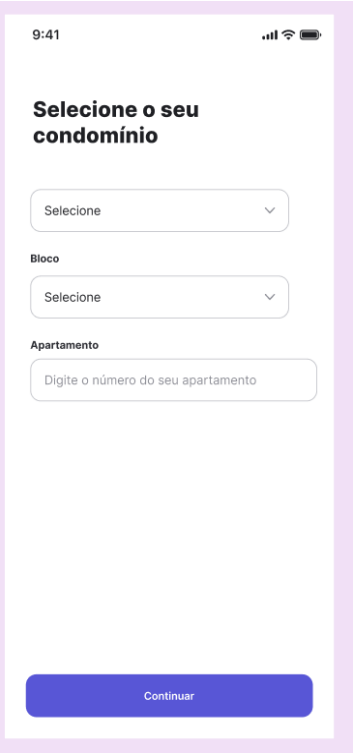
Fonte: A Autora

Ainda na Figura 12, na terceira tela, o processo de registro do condomínio continua, pedindo detalhes adicionais como a nomenclatura dos blocos, quantidade de vagas por blocos, quantidade de vagas por morador e tipo de nomenclatura das vagas por morador. Um botão “Continuar” também está presente na parte inferior da tela. Após clicar no botão, estará na tela da Figura 14, se for condomínio de vagas mistas ou estará na Figura 15, se for condomínio de vagas determinadas.

Na tela da Figura 13, o morador escolherá dentro do campo a opção do condomínio em que mora, preenchendo também com as informações do seu bloco e apartamento. Dependendo da modalidade do estacionamento, aparecerá a Figura 15 ou 14. Se for estacionamento de vagas mistas, será redirecionado a tela da Figura 14 e se forem vagas determinadas será redirecionado para a tela da Figura 15.

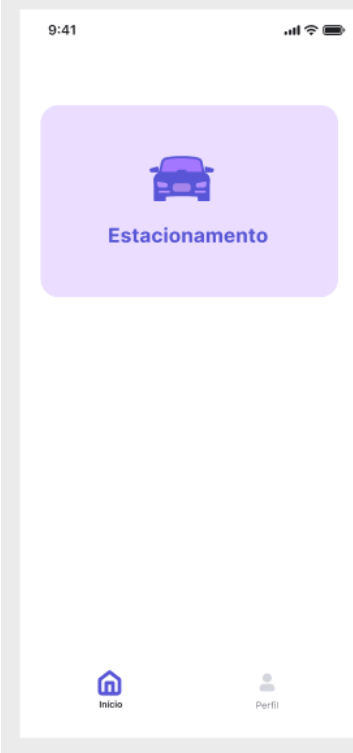
Tanto na Figura 14 quanto na 15, há um botão grande com a escrita “Estacionamento”. Ao clicar neste botão, o usuário será direcionado para diferentes telas dependendo do tipo de vaga que possui. Se for morador de um condomínio com vagas mistas, será redirecionado para a tela da Figura 16. Se, por outro lado, for morador de um condomínio com vagas determinadas, ele será direcionado para a tela da Figura 18. Na parte inferior de ambas telas de início, encontra-se uma barra de navegação *(tab bar)* com duas abas. A primeira aba corresponde à tela inicial, enquanto a segunda aba permitirá que o usuário visualize detalhes do cadastro, que será uma funcionalidade a ser desenvolvida futuramente. Na Figura 15 existe o botão “Negociar Minha Vaga”, ao clicar no botão aparecerá a tela da Figura 17.

Figura 13 - Tela de Selecionar o Condomínio



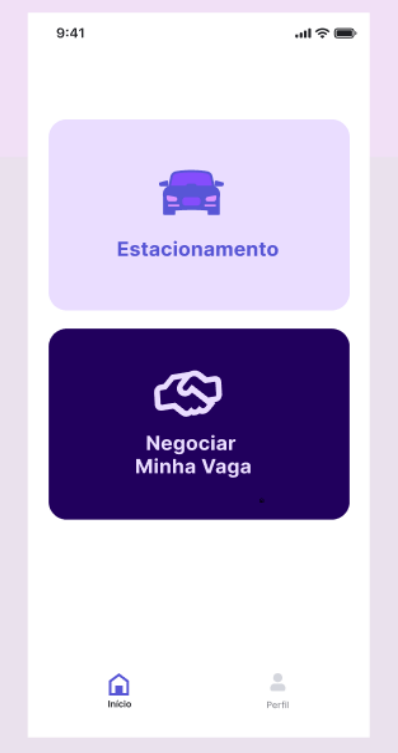
Fonte: A Autora

Figura 14 - Tela de Início de de condomínio de vagas mistas



Fonte: A Autora

Figura 15 - Tela de Início de condomínio de vagas determinadas



Fonte: A Autora

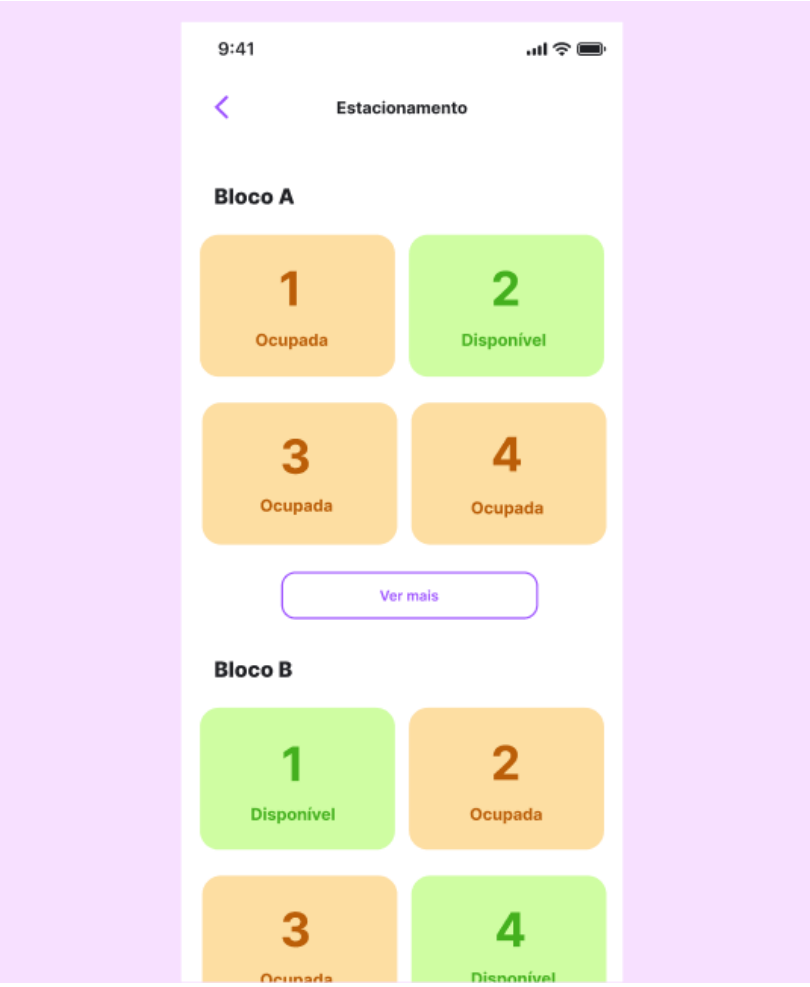
5.1.1. Estacionamento de Vagas Mistas

Esta seção mostra a tela de gestão de vagas de estacionamento mistas, escolhida no momento do cadastro de condomínio pelo síndico, Figura 12 - sabendo que para essa tela existir o condomínio deve fornecer imagem e vídeo da câmera, e o desenvolvedor fará o citado nas seções anteriores utilizando a visão computacional para executar as ações mostradas na Figura 3, 6 e 9. A funcionalidade funciona em tempo real, utilizando tecnologia de detecção de veículos, como mostrado nas Figuras 3, 6 e 8 e atualização contínua no banco de dados mostrada na Figura 9.

A interface da Figura 16 é composta por um texto no canto superior “Bloco A” uma grade de *cards* retangulares, cada um representando uma vaga de estacionamento específica. A disposição das vagas é organizada em duas colunas, com quatro vagas em cada bloco, podendo serem carregadas mais vagas se clicar no botão “Ver mais”. Cada *card* é colorido para indicar o status da vaga: laranja indica que a vaga está ocupada com o texto “Ocupada” e o verde indica que a vaga está livre com o texto “Disponível”.

Figura 16 - Tela de Visualização em Tempo Real das Vagas

Fonte: A Autora



A principal característica desta tela é a atualização em tempo real das informações sobre a disponibilidade das vagas. Isso é possível graças à integração com o *OpenCV,* como detalhado nas seções anteriores, que faz a detecção de veículos, que atualiza o banco de dados do condomínio sempre que um carro é detectado em uma vaga específica. Essa funcionalidade garante que os usuários tenham acesso a informações precisas e atualizadas sobre a ocupação das vagas, facilitando a gestão e a negociação de estacionamento.

A implementação desta seção oferece diversos benefícios aos usuários, incluindo a eficiência, reduzindo o tempo gasto na busca por vagas disponíveis; a confiabilidade de informações precisas e atualizadas em tempo real e a facilidade de uso por ter interface intuitiva e de fácil navegação.

5.1.2. Estacionamento de Vagas Definidas

Esta seção mostra as telas de gestão de vagas de estacionamento definidas, escolhida no momento do cadastro de usuário do síndico, a funcionalidade de visualização e possibilidade de negociação das vagas livres.

A tela de visualização e negociação de vagas é fundamental para a gestão eficiente das vagas de estacionamento em condomínios com vagas determinadas. Ela proporciona aos moradores uma visão clara e organizada das vagas disponíveis e ocupadas, facilitando a rápida identificação de opções de estacionamento.

Figura 17 - Tela de habilitar a permissão para negociar as vagas

Fonte: A Autora

Após o clique no botão “Negociar Minha Vaga” da Figura 15, aparecerá a tela da Figura 17. Esta tela tem como elementos principais o nome do bloco e número do apartamento, as vagas e dentro de cada vaga existe um botão de permissão, no qual que se ativá-lo, a vaga aparecerá como disponível para negociação na tela da Figura 18. Ao clicar em permitir, mostrará o campo de “Valor da vaga” para preencher com o preço mensal que deseja negociar com o vizinho do condomínio que tiver interesse. Se o botão de permissão estiver desativado, o campo de valor não aparece. Após as alterações terem sido realizadas, o botão “Salvar alterações” no canto inferior será clicado para gravar e ficar atualizada a informação da vaga na tela da Figura 18.

Na Figura 18, observa-se a organização das vagas de estacionamento disponíveis e ocupadas por cada morador. A interface encontra-se dividida em seções, cada uma representando um bloco específico, como o "Bloco A" e o "Bloco B". Dentro de cada bloco, as vagas estão numeradas, indicando os números dos apartamentos.

Figura 18 - Tela de Visualização e Negociação das Vagas

Fonte: A Autora

Para cada apartamento, há duas opções de vaga: “Esquerda” e “Direita”. No apartamento "1", a opção à esquerda está marcada como "Disponível", com um ícone de *check* verde, enquanto a opção à direita está marcada como "Ocupada", com um ícone de carro amarelo. A vaga do lado esquerdo possui o botão "Negociar", que possibilita o início da negociação da vaga e também possui o valor de R$150,00 mensais. Na parte inferior da tela, há um botão “Ver mais”, que permite ao usuário visualizar mais opções de vagas ou informações adicionais.

Enquanto no apartamento "2", tanto a opção à esquerda quanto à direita estão marcadas como "Disponível", cada uma com seu respectivo ícone de *check* verde e o botão "Negociar", com seus respectivos valores mensais. O botão "Negociar" desempenha um papel crucial na gestão dinâmica das vagas de estacionamento. Essa funcionalidade permite que os moradores iniciem negociações para utilizar vagas disponíveis. Ao clicar, será redirecionado para a tela da Figura 19, que possui dois botões “Cancelar” e “WhatsApp”, ao clicar em “WhatsApp” entrará

numa conversa dentro do aplicativo *WhatsApp* com o número da proprietária do apartamento para negociar. Ao clicar em “Cancelar”, retornará a tela da Figura 18.

Figura 19 - Tela de Ação de Negociação das Vagas

Fonte: A Autora

O elemento "Disponível" indica que a vaga está livre e pode ser negociada com outro morador. A presença do ícone de *check* verde facilita a identificação rápida das vagas livres, proporcionando uma visualização clara para os usuários. Já o elemento "Ocupada" indica que a vaga está em uso, seja por outro morador ou pelo próprio proprietário. O ícone de carro amarelo diferencia visualmente essas vagas das disponíveis, tornando mais fácil a distinção entre os espaços ocupados e livres no estacionamento.

A configuração das vagas de estacionamento, conforme ilustrado, simula um cenário onde o síndico cadastrou duas vagas por morador, sendo uma à direita e outra à esquerda. No entanto, essa configuração pode ser alterada dependendo das necessidades e do cadastro realizado pelo síndico de cada condomínio.

6. Conclusão

Por meio do desenvolvimento do aplicativo, foi possível concluir que existe viabilidade técnica no uso de Visão Computacional para melhorar a gestão de vagas de estacionamento em condomínios residenciais. As técnicas aplicadas, como

a detecção de ocupação de vagas por meio de reconhecimento de imagem em tempo real e a integração com a interface do *React Native* para a negociação de vagas, demonstraram que a solução pode trazer benefícios tanto para os moradores quanto para a administração do condomínio. No entanto, para validar completamente a eficácia do aplicativo, é necessário realizar testes adicionais, como testes de usabilidade com os residentes e avaliações em cenários de diferentes tipos de condomínios, garantindo a robustez e a eficiência da solução proposta.

A solução desenvolvida responde diretamente aos desafios mencionados na introdução, tanto para condomínios com vagas de uso indeterminado quanto para aqueles com vagas determinadas. Em ambos os cenários, demonstrou ser uma ferramenta que tem potencial para resolver os problemas do mundo real entre os residentes e otimizar o uso das vagas, minimizando os transtornos causados pela ausência de um sistema adequado de controle.

Contudo, devido ao fato dos testes terem sido realizados apenas em um ambiente controlado, ainda surgem desafios no que se refere à escalabilidade da solução em cenários de produção. Para sua implementação em condomínios de grande porte, com alta rotatividade de veículos e um número elevado de usuários, será necessária a realização de ajustes técnicos a fim de garantir que o sistema mantenha um desempenho eficiente em situações mais complexas.

Adicionalmente, foi identificada a oportunidade de incluir novas funcionalidades, como a possibilidade de os moradores atualizarem suas informações diretamente no aplicativo, o que contribuiria para aprimorar a experiência do usuário e facilitar a gestão de dados.

Dessa forma, conclui-se que o aplicativo apresenta grande potencial como uma solução inovadora e tecnicamente viável para estacionamento em condomínios, promovendo maior transparência, organização e flexibilidade. Entretanto, para que sua implementação em larga escala seja bem-sucedida, serão necessários aprimoramentos contínuos e testes futuros em ambientes reais. Com os devidos ajustes, o aplicativo pode se tornar uma ferramenta essencial para a administração de condomínios, beneficiando tanto moradores quanto administradores.

7. Considerações finais e trabalhos futuros

O projeto demonstrou o potencial de Visão Computacional para melhorar a gestão de vagas em condomínios residenciais, mas sua implementação em larga escala requer aprimoramentos. Embora tenha sido validada a viabilidade técnica e a usabilidade em um ambiente controlado, desafios como a escalabilidade em cenários mais complexos ainda precisam ser abordados. A partir das observações feitas durante o desenvolvimento, identificamos uma série de oportunidades para futuras evoluções.

Nos trabalhos futuros, será essencial focar na realização de testes em ambientes reais com diferentes tipos de condomínios para avaliar o desempenho do sistema em situações de alta demanda e com um número elevado de usuários. A ampliação de funcionalidades, como a possibilidade de os moradores atualizarem

suas informações, está planejada para aumentar a flexibilidade e melhorar a experiência do usuário. Além disso, a integração com sistemas de gestão de condomínios poderá proporcionar uma solução mais completa, unindo a gestão de vagas com outras funcionalidades de administração condominial.

Outro aspecto relevante para trabalhos futuros será a melhoria da segurança de dados e a implementação de notificações em tempo real, garantindo que o aplicativo atenda aos padrões de segurança e ofereça uma experiência fluida e eficiente para todos os usuários. Com esses avanços, tem o potencial de se consolidar como uma ferramenta robusta para a administração de estacionamentos em condomínios, beneficiando tanto moradores quanto administradores.

Referências

AGARWAL, Hitesh. *Advantages and Disadvantages of Using React Native*. 2023. Disponível em: https://techexactly.com/blogs/advantages-and-disadvantages-of-using-react-native. Acesso em: 24 set. 2024.

BRASIL. Lei n.º 12.607, de 3 de abril de 2012. Dispõe sobre o Sistema Nacional de Atendimento Socioeducativo e dá outras providências. Diário Oficial da União, Brasília, DF, 4 abr. 2012. Disponível em: https://legislacao.presidencia.gov.br/atos/?tipo=LEI&numero=12607&ano=2012&ato=661ATRU1kMVpWTe5d. Acesso em: 6 set. 2024.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, San Diego. p. 886-893.

DATE, C. J. An Introduction to Database Systems. 8. ed. Boston: Addison-Wesley, 2004.

DORNAIKA, F.; EL BAZ, A. Object Tracking and Motion Analysis Algorithms: A Comparison Using Real and Simulated Sequences. *Computer Vision and Image Understanding*, v. 1, n. 2, p. 221-236, 2020.

DUBois, P. MySQL Cookbook. 3. ed. Sebastopol: O'Reilly Media, 2013.
KALLAS, D.; THOMPSON, M. Database Systems: The Complete Book. 2. ed. New Jersey: Pearson, 2018.

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. 4. ed. Nova York: Pearson, 2018.

KAMIENSKI, Norbert. React Native Pros and Cons. 2023. Disponível em: https://pagepro.co/blog/react-native-pros-and-cons/. Acesso em: 24 set. 2024.

LU, W.; PLATEK, Z. Computer Vision-Based Vehicle Detection and Tracking in Parking Lots. *Journal of Intelligent Transportation Systems*, v. 23, n. 4, p. 345-359, 2019.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. Database System Concepts. 7. ed. Nova York: McGraw-Hill, 2019.
VAIDHEESWARAN, K.; GUPTA, R. Mastering MySQL for Web. Nova York: Packt, 2019.

SOUZA, Wellington Isac. Visão Computacional com Python - Crie 10 Projetos Incríveis. Udemy, 2024. Disponível em: https://www.udemy.com/course/visao-computacional-com-python/?couponCode=KEEPLEARNING. Acesso em: 24 set. 2024.

SZELISKI, R. Computer Vision: Algorithms and Applications. 2. ed. Nova York: Springer, 2021.

VILLAN, Alberto Fernandez. Mastering OpenCV 4 with Python. 2019. Disponível em: https://www.oreilly.com/library/view/mastering-opencv-4/9781789344912/79f055de-9d2a-4ec0-8009-3d2dfbe3080e.xhtml. Acesso em: 24 set. 2024.