**UNIVERSIDADE DE SOROCABA**

**PROJETO INTEGRADOR**

**ENGENHARIA DA COMPUTAÇÃO**

**Gabriel Pedrosa de Faria**

**Giovane da Silva Oliveira**

**Guilherme Bueno**

**Kathy Yuri Shimazu Otsubo**

**Sara Rayane Rodolpho**

***StopFinder***

**Sorocaba/SP**

**2024**

**Gabriel Pedrosa de Faria**

**Giovane da Silva Oliveira**

**Guilherme Bueno**

**Kathy Yuri Shimazu Otsubo**

**Sara Rayane Rodolpho**

**StopFinder**

Projeto Integrador do curso de Engenharia da Computação da Universidade de Sorocaba (UNISO)

Orientador: Prof. Gustavo Venancio Luz

**Sorocaba/SP**

**2024RESUMO**

**Palavras-chave:**

**ABSTRACT**

**Keywords:**

**LISTA DE ILUSTRAÇÕES**

[Figura 1 Representação detalhada de um bloco 10](#_Toc179277838)

[Figura 2 Estrutura geral do estacionamento 10](#_Toc179277839)

**SUMÁRIO**

[1 INTRODUÇÃO 6](#_Toc179277879)

[2 DESENVOLVIMENTO 7](#_Toc179277880)

[2.1 FRONT-END 7](#_Toc179277881)

[2.1.1 Tecnologias Utilizadas: 7](#_Toc179277882)

[2.1.2 Desenvolvimento Atual: 7](#_Toc179277883)

[2.1.3 Funcionalidades a Desenvolver: 8](#_Toc179277884)

[2.2 BACK-END 8](#_Toc179277885)

[2.2.1 Funcionalidades Implementadas: 8](#_Toc179277886)

[2.2.2 Implementação do código 9](#_Toc179277887)

[2.3 TAREFAS FUTURAS: 11](#_Toc179277888)

[3 Considerações Finais 12](#_Toc179277889)

[Referências 13](#_Toc179277890)

[anexo a – exemplo 14](#_Toc179277891)

# INTRODUÇÃO

Nos últimos tempos, o crescimento das frotas de veículos, vem aumentando o desafio de se locomover nas cidades urbanas. Entre os maiores problemas está a dificuldade de encontrar uma vaga de estacionamento, especialmente em áreas movimentadas. motoristas enfrentam dificuldades e uma longa busca por uma vaga, o que causa frustração, perda de tempo e contribui para o aumento do tráfego nas grandes cidades.

O ***StopFinder*** surge como uma solução para esse problema. Com o auxílio de tecnologia baseada em grafos, o aplicativo facilita a localização de vagas dos estacionamentos livres em tempo real. O algoritmo implementado no aplicativo não considera apenas a disponibilidade da vaga, mas também evita a repetição de mostrar a mesma vaga para diferentes pessoas, evitando aglomeração e transtornos.

Além de pensar nos motoristas, o ***StopFinder*** também oferece serviços para administradores de estacionamento, dando controle e monitoramento de vagas de maneira eficiente

O aplicativo ***StopFinder*** visa na melhoria de estacionar mais perto do seu destino final, e a redução de tempo de busca por vagas, que resulta na diminuição de congestionamento nas vias urbanas. Ao mesmo tempo, permite uma gestão mais eficiente dos estabelecimentos. O projeto se destaca como uma ferramenta essencial e fundamental para o mundo urbano e moderno em que vivemos hoje.

# DESENVOLVIMENTO

## FRONT-END

### Tecnologias Utilizadas:

Framework: Flutter

O Flutter foi escolhido para o desenvolvimento do front-end devido à sua capacidade de criar interfaces de usuário nativas para iOS e Android com uma única base de código. Oferece uma rica coleção de widgets e uma performance fluida.

Linguagem de Programação: Dart

Dart é a linguagem utilizada pelo Flutter, oferecendo uma sintaxe moderna e eficiente para o desenvolvimento de aplicações móveis.

Desenvolvimento Inicial: Canva

O design do front-end foi inicialmente desenvolvido no Canva, o que permitiu uma visualização preliminar das páginas e o planejamento da interface antes da implementação no Flutter.

### Desenvolvimento Atual:

* Página de Login: Tela onde os usuários podem fazer login no aplicativo.
* Cadastro: Tela para registro de novos usuários e fornecedores.
* Cadastro de fornecedor e/ou usuário: Páginas específicas para o cadastro de diferentes tipos de usuários (usuários e fornecedores).
* Homepage: Tela principal do aplicativo, apresentando opções e informações essenciais.
* Escolha do Local: Interface para selecionar o local onde o aplicativo será utilizado, com opções já implementadas para shopping, mercado e estacionamento.

### Funcionalidades a Desenvolver:

* Gerenciamento de Vagas: Página que exibirá a quantidade de vagas disponíveis para o gerenciamento do estabelecimento.
* Página de Dúvidas: Seção destinada a esclarecer possíveis dúvidas dos clientes.
* Página do Mapa: Tela que guiará o cliente até a vaga disponível, com funcionalidades de navegação e localização.

## BACK-END

Linguagem de Programação: Python

Bibliotecas Utilizadas:

**NetworkX:** Utilizada para criar e manipular grafos, permitindo a modelagem da rede de vagas e blocos.

**String:** Utilizada para criar nomes e identificadores dos blocos e vagas, facilitando a organização e referência no código.

### Funcionalidades Implementadas:

**Algoritmo de Encontrar Vaga Livre:**

* Critérios de Seleção: A função verifica se a vaga não está entre os 3 últimos lançamentos e se não está marcada como ocupada.
* Atualização de Lançamentos: A vaga selecionada é acionada como um dos últimos lançamentos, enquanto o antepenúltimo lançamento é removido da lista, mantendo sempre um histórico dos últimos 3 lançamentos.

**Criação de Grafos:**

* Armazenamento de Grafos: Criação de variáveis para armazenar a estrutura dos grafos que representam as vagas e blocos.
* Definição de Blocos e Nós: Cada vaga é representada como um nó, e blocos inteiros também são representados como nós. As arestas representam as distâncias entre os nós e a entrada.
* Distância entre Blocos: Calcula a distância da entrada até cada bloco e entre os blocos para auxiliar na busca e localização de vagas.

**Gestão de Blocos:**

* Criação de Identificadores: Utilização de strings para gerar identificadores, como letras (por exemplo, bloco A).
* Listagem e Armazenamento: Criação de uma lista de blocos que armazena o nome do bloco e as vagas pertencentes a cada bloco. Isso é implementado usando um dicionário para associar blocos a suas vagas.
* Adição de Vagas: Implementação de um loop para adicionar vagas aos blocos corretos e atualizar a lista de blocos com novos dados.
* Função para Encontrar Vaga Livre: Implementação de uma função específica para buscar vagas disponíveis com base nos critérios definidos e atualizar a lista de lançamentos.

O Grafo que representa as conexões entre as vagas e os blocos, e os blocos com a entrada.

### Implementação do código

Este sistema tem como objetivo gerenciar um estacionamento utilizando a tecnologia de grafos para organizar e localizar vagas disponíveis de forma eficiente. O Firebase é utilizado para armazenar informações como blocos de vagas, ocupação e dados dos usuários, enquanto o NetworkX é a biblioteca usada para a criação e manipulação dos grafos.

Na estrutura do sistema, o estacionamento é representado como um grafo onde os nós são as vagas e os blocos, e as arestas representam as conexões entre eles. A "Entrada" do estacionamento está conectada aos diferentes blocos (como Bloco A, Bloco B, e Bloco C), e cada bloco contém várias vagas, que também estão interconectadas. O sistema busca sempre a vaga disponível mais próxima, otimizando o processo de alocação para o motorista e para o administrador do estacionamento.

**representação visual do grafo:**

Esta imagem mostra como as vagas estão organizadas dentro de um bloco (neste caso, o Bloco A). Cada nó azul representa uma vaga, e as conexões entre elas indicam sua proximidade. A estrutura segue um padrão organizado, facilitando a busca de vagas pelo sistema.

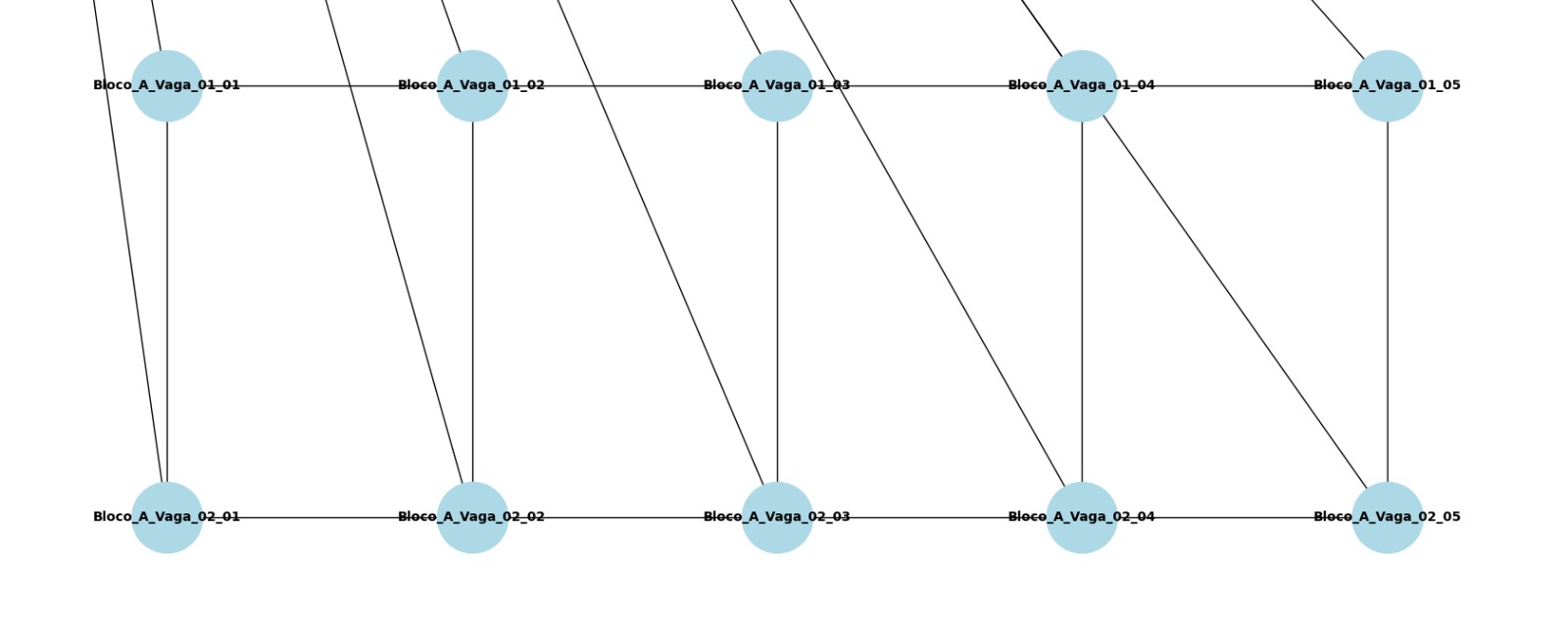


Figura 1 Representação detalhada de um bloco

Nesta imagem, temos uma visão completa do estacionamento. A “entrada” está conectada aos blocos A, B e C, que por sua vez estão ligados às vagas internas de cada bloco (como mostra o exemplo acima). Essa estrutura em grafo permite que o sistema encontre a vaga livre mais próxima de maneira eficiente, levando em consideração a distância entre os blocos e as vagas.

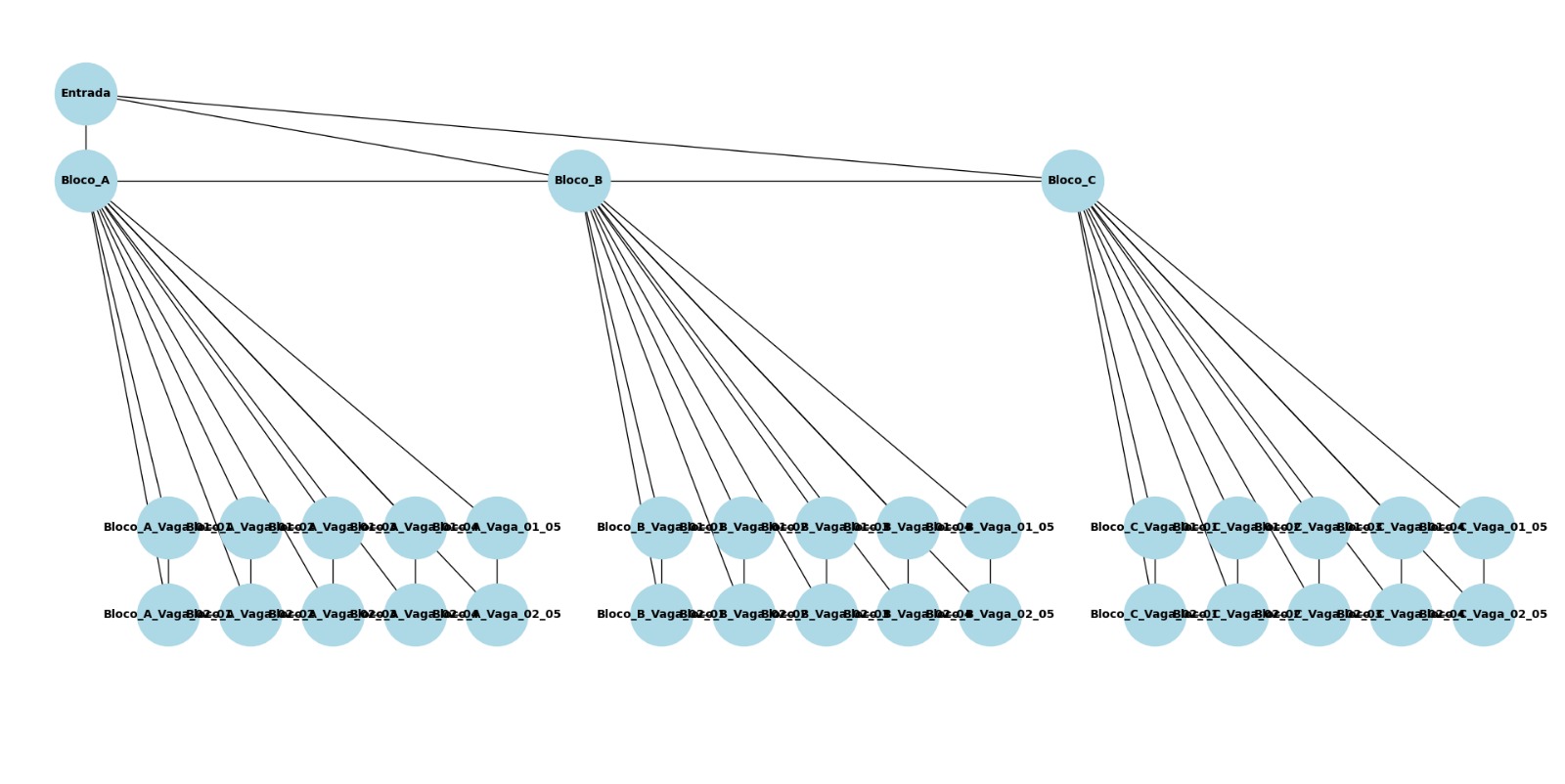


Figura 2 Estrutura geral do estacionamento

Com essa abordagem visual e baseada em grafos, o sistema oferece uma maneira inteligente de gerenciar a ocupação das vagas, evitando a alocação repetida das mesmas vagas recentemente ocupadas e otimizando a experiência tanto para motoristas quanto para administradores. A comunicação entre o sistema e o usuário é realizada via Firebase, garantindo que as informações sejam atualizadas em tempo real.

## TAREFAS FUTURAS:

* Integração de Sensores: Adição de sensores para validação automática das vagas. O sensor será integrado para realizar algumas validações de forma automática, melhorando a precisão e eficiência do sistema.
* Finalização das telas no Flutter
* Otimização e finalização do código em Python, para por fim, integrar o código no aplicativo finalizado.

## ANÁLISE DE CASOS DE USO

|  |  |
| --- | --- |
| **Situação** | **Descrição** |
| **Pior caso** | O usuário não conhece o aplicativo StopFinder e estaciona longe do destino desejado, sem qualquer otimização na escolha da vaga. |
| **Melhor caso** | O usuário conhece e utiliza o aplicativo StopFinder, conseguindo estacionar no bloco correto mais próximo ao destino. Se a vaga estiver ocupada, o app indica a vaga livre mais próxima. |

# Considerações Finais

Referências

CANVA. Disponivel em: <https://www.canva.com/pt\_br/>. Acesso em: 01 set. 2024.

AN ROSSUM, Guido; et al. Python. Disponível em: <https://www.python.org/>. Acesso em: 1 set. 2024.

GOOGLE. Flutter: UI toolkit for crafting natively compiled applications. Disponível em: <https://flutter.dev/>. Acesso em: 1 set. 2024.

GOOGLE. Firebase: Platform for building mobile and web applications. Disponível em: <https://firebase.google.com/>. Acesso em: 1 set. 2024.

anexo a – exemplo

Link das telas no canva:

<https://www.canva.com/design/DAGPJ0lbk5c/-ZMR7w_oTWhaV7pqHUu6Xg/edit?utm_content=DAGPJ0lbk5c&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>