**Documentação do Gerador de Rotas Otimizadas**

**Introdução**

Este documento tem como objetivo fornecer uma visão geral do software gerador de rotas, detalhando suas funcionalidades, arquitetura e como utilizá-lo.

**Funcionalidades**

* **Importação de dados:** O sistema aceita uma lista de coordenadas em formato JSON como entrada.
* **Otimização de rotas:** Utiliza dois algoritmos para otimizar as rotas:
  + **Algoritmo exaustivo:** Analisa todas as combinações possíveis, buscando a rota mais curta.
  + **Algoritmo heurístico:** Após 5 segundos, caso o algoritmo exaustivo não tenha encontrado a solução, um algoritmo heurístico é utilizado para encontrar uma solução aproximada, baseada na distância entre os pontos mais próximos.
* **Interface gráfica:** Uma interface simples e minimalista, desenvolvida com o framework Flet e a linguagem Python, permite a interação com o sistema e a visualização dos resultados.

**Benefícios**

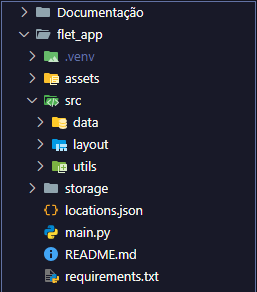
* Redução de emissão de gases poluentes em razão de deslocamentos menores.
* Aumento de produtividade visto que será possível economizar tempo nos trajetos e no planejamento das agendas, eliminando erros humanos.
* Economia de recursos, será possível reduzir o gasto com manutenção dos carros e a queima de combustíveis.

**Arquitetura**

* **Frontend:** Interface gráfica desenvolvida com o framework Flet, responsável pela interação com o usuário e visualização dos resultados.
* **Backend:** Lógica de negócio desenvolvida com a linguagem Python, responsável por:
  + Receber ou criar a lista de coordenadas em JSON.
  + Executar os algoritmos de otimização.
  + Retornar a rota otimizada para o frontend.

**Estrutura do Projeto**

O software inicialmente foi pensando para ser um serviço acessado por outros sistemas, desenvolvi a interface gráfica para deixar funcional para os usuários interessados utilizar, o programa sofreu várias refatorações para que a manutenção do mesmo seja viável. Segue estrutura de pastas:

* main.py: É o ponto inicial do sistema, responsável por guardar algumas configurações e chamar as funções necessárias.
* src>data: Guarda a lógica de interação com os dados e operação do arquivo JSON.
* src>layout: Contém os módulos referentes à interface gráfica.
* src>utils: Guarda a lógica e o módulo dos algoritmos geradores de rotas.
* requirements.txt: Gerencia as dependências do projeto, podendo ser facilmente reproduzido em outros ambientes.

**Fluxo de Trabalho**

1. **O usuário:**
   * Opera a lista de coordenadas pela interface gráfica podendo incluir novos pontos de parada.
   * Seleciona os pontos de parada que quer incluir na rota.
   * Clica no botão "Gerar Rota".
2. **O sistema:**
   * Lê o arquivo JSON e extrai as coordenadas.
   * Inicia o algoritmo exaustivo.
   * Após 5 segundos, caso não tenha resolvido, interrompe o algoritmo exaustivo e inicia o algoritmo heurístico.
   * Retorna a rota otimizada para o frontend.
3. **O frontend:**
   * Exibe a rota otimizada em uma lista contendo os links integrados já com o Google Maps.

**Algoritmos de Otimização**

* **Algoritmo exaustivo:**
  + **Descrição:** Analisa todas as possíveis combinações de rotas, calculando a distância total de cada uma.
  + **Complexidade:** Crescimento exponencial com o número de pontos, tornando-o inviável para grandes conjuntos de dados.
* **Algoritmo heurístico:**
  + **Descrição:** Ordena os pontos pela distância entre eles e constrói a rota conectando os pontos mais próximos.
  + **Complexidade:** Mais eficiente para grandes conjuntos de dados.

**Futuras implementações**

* Criação de um input para carregar o arquivo JSON de coordenadas, permitindo a importação de arquivos externos.
* Criação de uma forma para inserir o ponto de início e fim da rota, podendo ser capturado a localização atual do dispositivo utilizado.
* Implementação da paralelização para melhoria de desempenho.
* Deploy em sistemas Android, IOS e Web.