UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

SISTEMAS DISTRIBUÍDOS

VINICIUS DOS REIS LEAL
NICOLLAS KAUÃ PEREIRA REIS

01/11/2024

CUIABÁ - MT

Documentação do Projeto de Previsão Meteorológica Distribuído

Visão Geral

Este projeto é uma aplicação distribuída para previsão do tempo em uma cidade específica, implementando um servidor e cliente com gRPC e uma API web com Flask. Ele conecta-se ao serviço de previsão AccuWeather para obter dados meteorológicos e oferece uma interface de consulta para usuários finais.

Arquivos e Estrutura.

1. Cliente.py

Descrição: O arquivo cliente.py implementa o cliente que se comunica com o servidor gRPC para requisitar as previsões meteorológicas.

Função Principal:

- O cliente estabelece uma conexão com o servidor gRPC na porta 50051.
- Envia uma solicitação contendo o nome da cidade desejada.
- Recebe uma resposta com uma lista de previsões para os próximos dias e exibe os detalhes no console.

Como Funciona:

- O cliente usa uma interface gerada a partir do arquivo **.proto**, permitindo que ele envie dados e receba respostas estruturadas do servidor.
- Ele lida com a serialização e deserialização automaticamente, graças ao protocolo gRPC.

2. servidor.py

Descrição: O arquivo **servidor.py** implementa o servidor gRPC que processa as requisições de previsão e se conecta à API AccuWeather para buscar os dados de previsão.

Função Principal:

- Recebe requisições de clientes gRPC contendo o nome de uma cidade.
- Conecta-se à API AccuWeather e busca dados de previsão para a cidade especificada.
- Retorna uma lista de previsões, incluindo informações como data, condição climática, temperatura máxima, sensação térmica e umidade.

Detalhes do Processo:

- A comunicação segue o formato definido no arquivo .proto, o que facilita a compatibilidade entre cliente e servidor.

- O servidor é executado na porta 50051, mantendo uma conexão aberta para receber requisições de clientes.

3. previsao.proto

Descrição: O arquivo previsao.proto define o serviço e as mensagens para o gRPC.

Componentes do Arquivo:

Serviço: Define PrevisaoService, com o método GetPrevisao, que solicita o nome da cidade e retorna uma lista de previsões.

Mensagens:

- PrevisaoRequest: Contém o campo cidaade, que representa o nome da cidade para a previsão.
- PrevisaoResponse: Contém uma lista de previsões, cada uma detalhada com data, temperatura máxima, sensação térmica, umidade e condição climática.
- **Importância**: Este arquivo permite que cliente e servidor compartilhem uma estrutura comum de comunicação, evitando erros e incompatibilidades.

4. web/api.py

Descrição: Implementa uma API web com Flask para fornecer uma interface HTTP para as previsões meteorológicas.

Função Principal:

- A API possui uma rota /previsao que recebe o nome da cidade como parâmetro.
- A rota chama o servidor gRPC para buscar as previsões e retorna os dados em formato JSON, tornando-o acessível via HTTP.

Como Funciona:

- Flask escuta na porta HTTP, oferecendo uma interface para consultas REST.
- O cliente HTTP pode fazer uma requisição com o nome da cidade e obter a resposta em JSON, facilitando a integração com interfaces web ou aplicativos móveis.

5. Arquivos de Interface (index.html e styles.css)

Descrição: Estes arquivos compõem a interface de usuário para interação com a API de previsão meteorológica.

- **index.html**: Oferece uma interface simples para o usuário inserir o nome de uma cidade e consultar a previsão.

- **styles.css**: Adiciona estilos visuais ao HTML, garantindo uma experiência de usuário mais agradável.

6. requirements.txt

Descrição: Este arquivo lista as dependências do projeto, permitindo que o ambiente seja configurado de maneira rápida e consistente.

Bibliotecas Principais:

- Flask: Framework web para a API.
- grpcio e grpcio-tools: Necessários para a implementação do servidor e cliente gRPC.
- Outras bibliotecas específicas para conexão e manipulação de dados.

Configuração e Execução

1. **Instalar Dependências**:

No terminal, navegue até a pasta do projeto e execute: <pip install -r requirements.txt>

2. **Gerar Código gRPC (se necessário)**:

Para regenerar os arquivos **previsao_pb2.py** e **previsao_pb2_grpc.py**, execute:

<python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. previsao.proto>

3. **Executar o Servidor**:

Inicie o servidor gRPC com o comando: <python servidor.py> ou <python3 servidor.py(Linux: Debian)>

4. **Iniciar a API Web**:

Em outro terminal, execute a API Flask: <python web/api.py> ou <python3 web/api.py>

5. **Executar o Cliente**:

Por fim, em um terceiro terminal, execute o cliente gRPC: <python cliente.py> ou <python3 cliente.py>