Fatec Ipiranga
Programação Avançada Orientada a Objetos
Professor Rodrigo Bossini
Especificação do projeto

Sobre ferramentas de Inteligência Artificial

O uso de ferramentas de inteligência artificial, como ChatGPT, Gemini, Github Copilot (deve ser desativado no IDE) ou qualquer outra é proibido. No dia a dia, é óbvio que profissionais de diferentes áreas as utilizam, visando produtividade. Entretanto, você está em um ambiente de aprendizado. Não é o momento de utilizá-las. Caso o seu projeto contenha qualquer indício de uso de tais ferramentas, a nota será zero. O grupo fica sujeito a processo administrativo que pode resultar em advertência formal.

Observação: Somente o conteúdo visto em aula e/ou mencionado neste documento pode ser utilizado. Caso a solução contenha qualquer conteúdo, mínimo que seja, em desacordo com esta regra, a nota será zero.

Instruções

- O projeto pode ser desenvolvido por grupos de até quatro alunos.

- Adicione um arquivo README.md, incluindo o nome completo e RA de cada integrante, em ordem alfabética.

- Datas
Desenvolvimento: 19/11
Apresentação: 26/11
Limite para entrega: Antes do início da aula de apresentação.

- A solução deve ser armazenada no Github de um membro do grupo. O repositório deve ser um fork do seguinte repositório.

https://github.com/professorbossini/20252_fatec_ipi_paoo.git

Trata-se do repositório utilizado pelo professor em aula. Embora ele possua todo o conteúdo do semestre, o trabalho envolve apenas o conteúdo sobre microsserviços.

- O grupo deve entregar o link de seu repositório preenchendo o Google Form a seguir.

https://bit.ly/bossini_fatec_entregas_provas

- O controle de versão seguindo boas práticas é obrigatório.

- O projeto deve ser feito em Javascript com NodeJS, a menos de eventuais sugestões/indicações presentes neste documento.

Requisitos

Classificação de lembretes

- Ajuste a solução para que lembretes também sejam classificados. Um lembrete pode ser importante ou comum. Todo lembrete que tem pelo menos 50 caracteres é importante. Os demais, são comuns. Lembre-se de trabalhar de modo que os microsserviços permaneçam altamente coesos, como visto em aula.

Perda de eventos a serem classificados

No momento, caso o microsserviço de classificação fique inoperante, ele perde eventos. Ajuste a solução para que ele lide com eventos potencialmente perdidos quando voltar a operar. Lembre-se que isso foi feito em aula, com o microsserviço de consulta.

Microsserviço de logs

Crie novo microsserviço. Ele deve possuir uma base local que armazena logs referentes a cada evento. Para cada evento, a sua base deve armazenar.

- Um identificador (UUID)
- Data e hora do sistema, extraída na hora em que ele receber uma requisição
- Tipo do evento
- Dados (que também chamamos de payload ou carga útil nas aulas) do evento

O microsserviço deve disponibilizar um endpoint /logs que permite a obtenção da sua base completa.

Microsserviço de estatística

Crie novo microsserviço. Ele deve possuir uma base local que armazena dados estatísticos a respeito de cada tipo de evento. Os dados devem ser os seguintes.

- total de lembretes
- total de lembretes comuns versus importantes
- total de observações criadas
- média de caracteres por observação

Os dados devem ser armazenados já no momento em que o microsserviço recebe um evento.

O microsserviço deve disponibilizar um endpoint /estatistica que permite a obtenção da sua base completa.

Barramento armazena eventos perdidos por tipo

Ajuste a base do barramento de eventos utilizada para o armazenamento de eventos perdidos. Agora, ela deve ser um objeto que possui, como chaves, todos os possíveis tipos de eventos. A cada chave, deve existir uma lista de eventos associada daquele tipo. Veja um exemplo.

```
{
 LembreteCriado: [
   {
     tipo: LembreteCriado,
     dados: {
       id: 1,
       texto: 'fazer café'
     }
   }
 ],
 ObservacaoClassificada: [
   {
     tipo: ObservacaoClassificada,
     dados: {
       idLembrete: 1,
       id: '4455abcefffsa',
       texto: 'comprar açúcar',
       status: 'comum'
     }
   }
 ]}
```

O barramento deve disponibilizar um endpoint /registrar que permite que um microsserviço indique quais eventos são de interesse para ele.

Exemplos

- O microsserviço de lembrete não se interessa por evento algum
- O microsserviço de logs se interessa por todos os tipos de eventos
- O microsserviço de classificação se interessa pelo evento ObservacaoCriada, apenas

Uma vez que esse endpoint seja criado, faça com que cada microsserviço, assim que entrar em execução, avise o barramento quais tipos de eventos lhe interessam. Cada microsserviço envia seu nome simples (lembretes, observações, consulta etc) e o barramento associa seus nomes com a lista de eventos de interesse de cada um.

Ao direcionar um evento para um microsserviço, o barramento verifica se o microsserviço se interessa por aquele evento.

Desafios (além da prova)

Cada desafio implementado vale 0,5 ponto na nota de grupo da P2.

Classificação com Inteligência Artificial

Faça com que ele lembretes e observações sejam classificados utilizando a API do Gemini: seu microsserviço de classificação envia um lembrete ou observação ao Gemini, lhe perguntando se, pelo texto, aquilo parece importante ou comum.

Veja um exemplo de interação com o Gemini usando Javascript.

```
import { GoogleGenAI } from "@google/genai";

const ai = new GoogleGenAI({});

async function main() {
  const response = await ai.models.generateContent({
    model: "gemini-2.5-flash",
    contents: "Esse lembrete parece importante? Comprar milho",
  });
  console.log(response.text);
}
```

Esse exemplo foi extraído da documentação oficial, que pode ser acessada por meio do link a seguir.

https://ai.google.dev/gemini-api/docs?hl=pt-br#javascript

Microsserviço de consulta com Python ou Java

Reescreva o microsserviço de consulta usando Python ou Java.

Se optar por Java, defina a API usando o Spring Boot. Veja um exemplo de endpoint definido com Java e Spring Boot no link a seguir.

https://spring.io/guides/gs/rest-service

Se optar por Python, defina a API usando FastAPI ou Django.

Veja um exemplo de endpoint definido em Python com FastAPI no link a seguir.

https://fastapi.tiangolo.com/tutorial/first-steps/

Veja um exemplo de endpoint definido em Python com Django.

https://docs.djangoproject.com/en/5.2/intro/tutorial01/