Solução: Projeto Watson

Autor: Gustavo F Penedo

Links úteis

Chatbot Preview

capbot API

Antes de começar

Essa solução foi criada em um ambiente conda com **Python 3.8**.

Tendo o pacote Anaconda instalado, execute os comandos abaixo para configurar o ambiente:

```
conda create -n capBot
conda activate capBot
conda install python=3.8
```

Com o ambiente instalado e ativado, insira o comando abaixo para instalar os pacotes necessários:

```
pip install -r requirements.txt
```

Feito isso, entre no diretório do projeto e execute o comando para inicializar a API:

```
python manage.py runserver
```

Se tudo estiver correto, o servidor estará disponível no endereço: **localhost:8000**

1. Introdução

Esse repositório busca demonstrar os meus conhecimentos em desenvolver um projeto Watson Assistant como implementar uma API para salvar ou ler suas mensagens. As seções abaixo esclarecem melhor as minhas escolhas para cada requisito solicitado.

2. Modelagem

2.1 Watson Assistant

Criou-se um chatbot na plataforma Watson que fosse capaz de traduzir uma interação simplificada com o usuário e seguisse a lógica proposta pelo problema.

O processo de fazer um pedido foi separado nas etapas para que o cliente possa escolher com cautela o pedido. Porém também foi garantido que o cliente possa montar o pedido com uma linha de texto.

Durante as etapas de montar o pedido também é possível conferir informações nutricionais de cada alimento.

Para saber mais sobre as rotinas do chatbot, confira a seção **4.1**

2.2 API

Para o desenvolvimento da API foi escolhido a framework **Django** junto com o pacote **Django Rest Framework** e **Django Filters** para entregar uma ferramenta com formato REST e possuir a habilidade de inserir filtros.

Por questão de segurança, foi pensado em autentificação por token em uma aplicação real. Entretanto não foi desenvolvido para descomplicar a reprodução do código.

Foi feito o deploy da API no serviço **Heroku** para que, de forma simplifica, fosse possível monitorar a atividade da aplicação bem como monitorar e reagir a falhas.

2.3 Banco de Dados

Foi escolhido o banco **PostgreSQL** para armazenar as informações salvas pela API. O banco está presente na plataforma **Heroku** que possui um serviço otimizado para tal.

3. Rotas da API

Conforme solicitado para a solução, os endpoints da aplicação ficaram como demonstrado abaixo.

- **Bots:**
  - GET /bots
  - GET /bots/:id
  - POST /bots/
  - PUT /bots/:id/
  - DELETE /bots/:id/
- **Mensagens:**
  - GET /messages
  - GET /messages/:id
  - GET /messages/?conversationId=:conversationId
  - POST /messages

4. Testes

4.1 Watson Asisstant

Foi determinado as seguintes rotinas de testes manuais, extraídos do enunciado do problema:

- Identificar uma pergunta nutricional
- Identificar um pedido
  - Escolher uma proteína e sua opção de preparo
  - Escolher um carboidrato e sua opção de preparo

- Escolher uma salada ou montar um mix de saladas
- Buscar informações nutricionais durante os 3 processos descritos acima
- Gravar endereço de entrega
- Gravar método de Pagamento
- Simular sistema de pagamento
- Concluir pedido

4.2 API

Foram feitos testes unitários em cada rota da API conforme mencionado na proposta do projeto.

Os testes podem ser encontrados na pasta de cada aplicação. **Ex: bots/test.py e bot_messages/test.py**

Para conferir os resultados, basta executar o comando abaixo dentro do diretório do projeto:

```
python manage.py test
```

Notas Finais

Qualquer dúvida com o projeto, favor entrar em contato.

Atenciosamente,
Gustavo F Penedo

Contato: gustavo-penedo@hotmail.com