Mirson Mboa

Desenvolvimento de uma aplicação com os conceitos abordados

Licenciatura em Informática

Universidade Save

Chongoene

2024

1. Introdução

Este guia apresenta os passos para criar um sistema web robusto com:

- Autenticação de dois fatores (2FA);
- Caching;
- Comunicação de aplicações usando APIS;
- Registo de auditoria para ações dos usuários;
- Testes unitários com Pytest e automação;
- Deploy no git.

2. Pré-requisitos

Antes de iniciar, certifique-se de ter os seguintes itens configurados:

2.1 Ferramentas

- Python 3.12;
- Gerenciador de pacotes pip;
- Ambiente virtual Python (venv recomendado);
- Banco de dados SQLite (ou outro compatível com SQLAlchemy).

2.2 Bibliotecas Necessárias:

- ✓ Flask;
- ✓ Flask-SQLAlchemy;
- ✓ Flask-Login;
- ✓ Flask-Bootstrap;
- ✓ Flask-WTF;
- ✓ WTForms;
- ✓ Language-Tool-Python;
- ✓ PyQRCode;

- ✓ PyPNG;
- ✓ OneTimePass;
- ✓ Requests;
- ✓ Werkzeug;
- ✓ Google-GenerativeAI.

2.2.2. Componentes do Sistema:

Sistema de Login Seguro:

- **Modelos de Usuário**: Gerencie usuários e senhas com hashing seguro (bcrypt).
- **Autenticação de Dois Fatores (2FA)**:
    - Geração de um código OTP (One-Time Password) único por usuário.
    - Geração de QR Code para configuração em aplicativos de autenticação (Google Authenticator).

Comunicação de aplicações usando APIS:

- Utilize a biblioteca google-generativeai para gerenciar conversas no chat.
- Respostas são corrigidas gramaticalmente pelo LanguageTool.

Registro de Auditoria:

- Todas as ações relevantes (login, logout, falhas) são registradas em logs no banco de dados.

Caching

LRU Cache:

- Utilizado para armazenar em cache respostas da API Google Gemini para perguntas frequentes.
- Implementado com o decorador `@lru_cache(maxsize=100)` para a função `get_gemini_response`.

Testes Unitários

- Utilizamos **Pytest,** para garantir que cada parte do sistema funciona como esperado.

Backend:

- **Flask Framework**: Para desenvolvimento do servidor e roteamento.
- **Flask SQLAlchemy**: Para gerenciamento do banco de dados.
- **Flask Login**: Para autenticação e gerenciamento de sessões de usuário.
- **Werkzeug Security**: Para hashing e validação de senhas.
- **Flask Bootstrap**: Para estilização das páginas.

Banco de Dados:

- **SQLite**: Banco de dados leve para armazenar usuários, logs de auditoria e informações relacionadas.

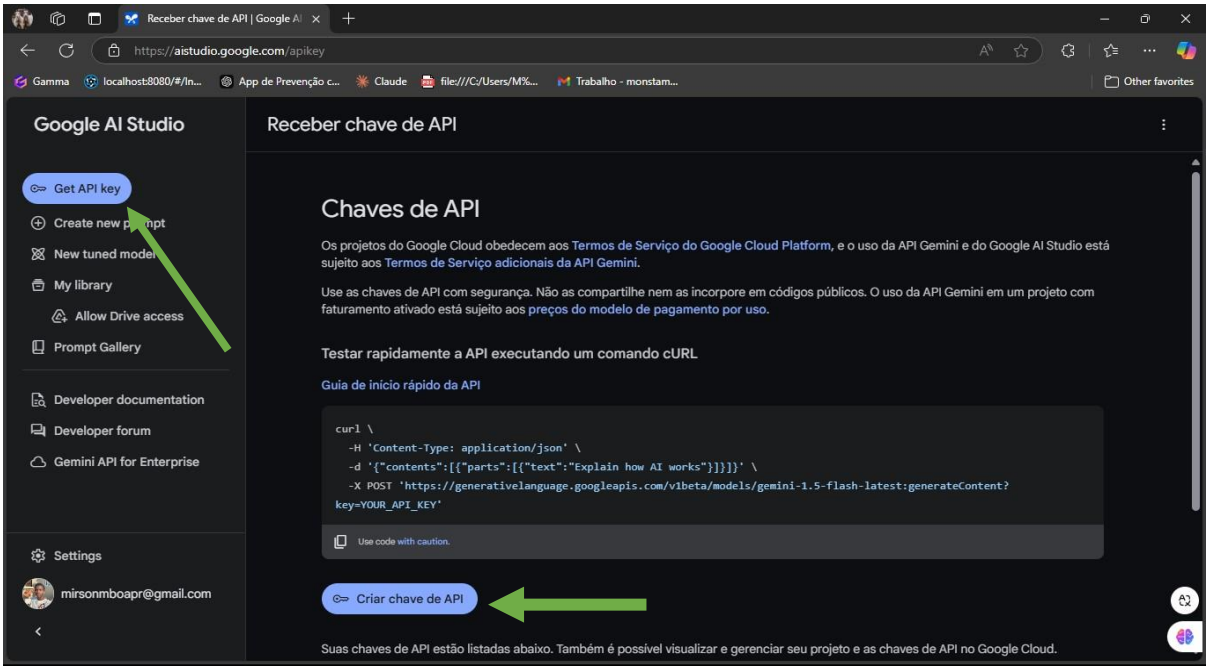
API de Linguagem Natural:

- **Google Gemini**: Modelo de IA para geração e compreensão de linguagem.
- Configuração feita com a chave de API.
- Utilização da biblioteca google.generativeai.

3. Obter e configurar a API do Google Gemini

Para obter e configurar a API do Google Gemini, siga estes passos:

- Acesse o Google AI Studio, através do link: https://aistudio.google.com
- Clique em **Get API Key** e de seguida, clique em **Criar Chave de API.** (Criar Projeto se for seu primeiro projeto)



Como configurar a API do Google Gemini:

No seu código Python, você deve definir a chave da API, através de uma variável:

GOOGLE_GEMINI_API_KEY = 'Sua_Chave_Aqui'

Usar a chave para configurar a API: Após configurar a chave, você deve inicializar a API do Google Gemini no seu código.

- Importe a biblioteca do Google Gemini:

```
import google.generativeai as lua
```

- Configurando a chave da API:

```
gemini.configure(api_key=GOOGLE_GEMINI_API_KEY)
```

Configurar o Modelo de IA:

Após a configuração da chave da API, você precisa configurar o modelo de IA do Google Gemini que será usado para responder às mensagens dos usuários. Isso é feito da seguinte maneira:

`model = gemini.GenerativeModel('gemini-1.5-pro-latest')`: Aqui você cria um objeto de modelo do Google Gemini. O parâmetro `'gemini-1.5-prolatest'` é o nome do modelo que você deseja usar.

`chat = model.start_chat(history= [])`: Isso inicializa a conversa com o modelo, começando com um histórico vazio.

Criar Funções de Processamento (Limpeza e Correção):

Limpeza da Resposta:

```
def limpar_resposta(resposta):
    return re.sub(r'[^\w\s+\-=\!?@#,.()*$%]', '', resposta)
```

Esta função remove caracteres especiais indesejados da resposta do modelo, mantendo apenas os caracteres válidos.