

SISTEMA DE GERENCIAMENTO ESPACIAL

Prof^a Cynthia Moreira Maia

DESENV. RÁPIDO DE APLICAÇÕES EM PYTHON





INTRODUÇÃO

O projeto “SpaceRecords” visa criar um sistema de gerenciamento de expedição espacial para facilitar o planejamento, monitoramento e execução de missões espaciais. Utilizaremos o método RAD (Rapid Application Development) para desenvolver o sistema. A linguagem escolhida é Python, com a biblioteca Flask para a construção da aplicação web e um banco de dados embutido SQLite para armazenar informações sobre as missões.

MODELAGEM DO NEGÓCIO



- IDENTIFICAÇÃO DE REQUISITOS
- PROTOTIPAGEM INICIAL
- DEFINIÇÃO DE CASOS DE USO
- VALIDAÇÃO COM STAKEHOLDERS
- BANCO DE DADOS EMBUTIDO
- MODELAGEM CONCEITUAL
- DEFINIÇÃO DE FLUXOS DE TRABALHO

MODELAGEM DOS DADOS



- MODELAGEM CONCEITUAL
- MODELAGEM LÓGICA
- NORMALIZAÇÃO
- INDEXAÇÃO
- CONSIDERAÇÕES DO DESEMPENHO

CONSTRUÇÃO RÁPIDA

Desenvolveremos funcionalidades incrementais para o nosso sistema de gerenciamento de expedição espacial. Vamos criar rotas em nossa aplicação Flask para realizar as operações CRUD (Create, Read, Update, Delete) relacionadas às missões espaciais.

- Rotas da Aplicação (Flask):

- a. Visualizar Todas as Missões
- b. Recuperar Detalhes de uma Missão Específica
- c. Pesquisar Missões por Intervalo de Datas
- d. Adicionar Nova Missão
- e. Editar Missão Existente
- f. Excluir Missão

- 3.2 Integração com o Banco de Dados SQLite:

- 3.3 Páginas HTML:

- 3.4 Testes

IMPLEMENTAÇÃO

Transformaremos nossos planos e conceitos em código real. Vamos criar a aplicação web usando o framework Flask, conectar ao banco de dados MySQL e desenvolver as funcionalidades que permitirão aos usuários gerenciar as missões espaciais. Aqui estão os detalhes:

- **Configuração do Ambiente:**

- Instalaremos o Python, o Flask.

- Configuraremos o ambiente de desenvolvimento.

- **Estrutura do Projeto:**

- Criaremos uma estrutura de diretórios para organizar nosso código:

- └── app/
 - └── static/
 - └── style.css
 - └── templates/
 - └── index.html
 - └── mission_details.html
 - └── edit_mission.html
 - └── __init__.py
 - └── routes.py
 - └── config.py
 - └── run.py
 - └── requirements.txt

- **Criação das Rotas:**

- No arquivo routes.py, criaremos as rotas para cada funcionalidade:

- o Visualizar todas as missões.

- o Recuperar detalhes de uma missão específica.

- o Adicionar nova missão.

- o Editar missão existente.

- o Excluir missão

IMPLEMENTAÇÃO

- **Integração com o Banco de Dados SQLite:**

- Configuraremos a conexão com o banco de dados SQLite no nosso aplicativo Flask.

- **Páginas HTML:**

- Criaremos páginas HTML na pasta templates para exibir as informações:

- o index.html: Lista de todas as missões.

- o mission_details.html: Detalhes de uma missão específica.

- o edit_mission.html: Formulário de edição para modificar os detalhes de uma missão.

- **Estilização:**

- Utilizaremos o arquivo style.css na pasta static para estilizar nossas páginas.

- **Testes:**

- Realizaremos testes para garantir que todas as funcionalidades estejam funcionando corretamente.

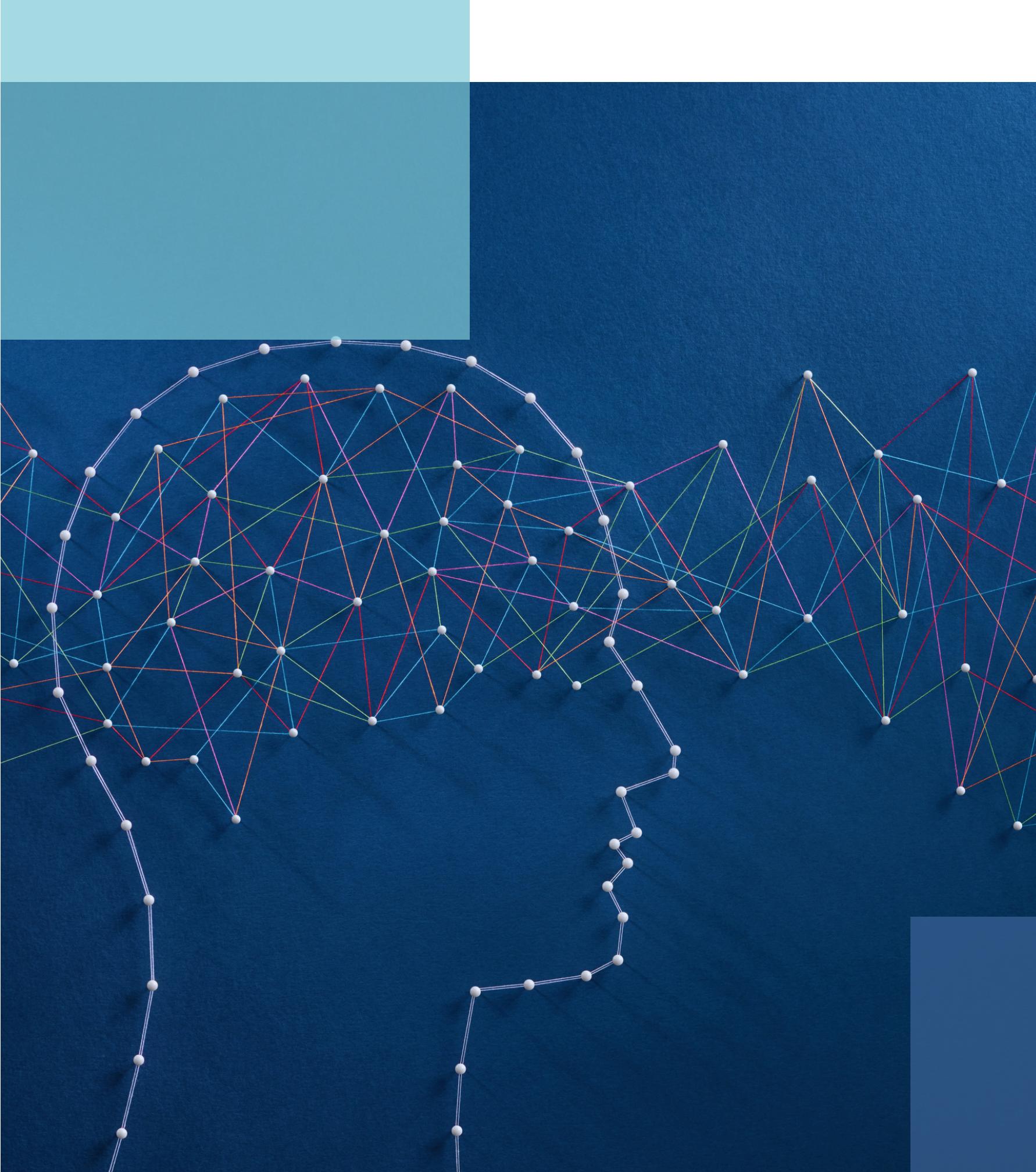
- Verificaremos se as operações CRUD estão sendo executadas conforme o esperado.

- **Deploy:**

- Quando tudo estiver funcionando localmente, podemos implantar nossa aplicação em um servidor web para que os usuários possam acessá-la.

CONCLUSÃO

Ao longo das etapas de planejamento, desenvolvimento e implementação, enfrentamos desafios complexos e colaboramos de forma eficiente para superá-los. Os resultados obtidos demonstram claramente o impacto positivo que essa iniciativa terá em nossa formação e na experiência.



OBRIGADO!

Atenciosamente toda equipe

Camile Lima, Rodrigo Lima, Christian Abreu, Carlos Victor, Mauricio Melo