Centro Universitário Estácio
Campus Parangaba

Disciplina:
Desenvolvimento rápido de Aplicações em Python

Alunos:
Rodrigo Lima e Sousa 202302569791,
Christian Abreu de Souza, 202308424009,
Carlos Victor Sousa de Oliveira, 202303147988,
Camile de Lima Fernandes, 202302333109,
Maurício de Melo Bezerra Filho 202303971435,

Professor: Cynthia Maia

Fortaleza-Ce
2024

Relatório de Gerenciamento de Expedição Espacial
Projeto "SpaceRecords"

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.

Fases do Projeto

Fase 1: Modelagem do Negócio

Aqui nos concentramo-nos em estabelecer os requisitos básicos do sistema e criar protótipos para entender melhor como ele funcionará. Principais passos:

1.1 Identificação de Requisitos:

Começamos coletando informações sobre o que o sistema deve fazer. Isso envolve conversas com os stakeholders, como astronautas, engenheiros, cientistas e administradores da missão. Perguntas importantes nesta fase incluem: Quais funcionalidades o sistema deve ter? Quais informações precisam ser rastreadas para cada missão? Como os usuários interagirão com o sistema?

1.2 Prototipagem Inicial:

Criaremos protótipos de telas ou fluxos de trabalho para visualizar como o sistema funcionará. Criar esboços de telas para adicionar uma nova missão, visualizar detalhes de uma missão existente e editar informações.

1.3 Definição de Casos de Uso:

Identificaremos os principais casos de uso do sistema. Isso inclui as principais tarefas que os usuários executarão. Alguns casos de uso relevantes para nosso projeto: Adicionar nova missão. Visualizar detalhes de uma missão. Editar informações de uma missão existente. Excluir uma missão.

1.4 Validação com Stakeholders:

Apresentaremos os protótipos e casos de uso aos stakeholders para obter feedback, é essencial. Isso nos ajuda a garantir que estamos atendendo às necessidades de todos os envolvidos.

1.5 Banco de Dados Embutido:

O SQLite é uma base de dados embutida, o que significa que não requer um servidor separado. O arquivo do banco de dados SQLite armazena todas as informações em um único arquivo. Não precisaremos configurar um servidor, em vez disso, nosso aplicativo Flask se comunicará diretamente com o arquivo SQLite.

1.6 Modelagem Conceitual:

Criaremos um modelo conceitual que representa as principais entidades e relacionamentos do sistema. No nosso caso, teríamos entidades como "Missão", "Tripulação" e "Carga Útil", com relacionamentos entre elas.

1.7 Definição de Fluxos de Trabalho:

Detalharemos como as informações fluirão pelo sistema. Por exemplo, quando uma nova missão é adicionada, como ela é registrada no banco de dados e como os usuários a visualizam posteriormente.

Fase 2: Modelagem do Dados

Criaremos representações estruturadas dos dados que serão armazenados e manipulados pelo nosso sistema de gerenciamento de expedição espacial. Essa etapa é crucial para garantir a integridade, consistência e eficiência do banco de dados. Vamos detalhar:

2.1 Modelagem Conceitual:

A modelagem conceitual envolve a criação de um modelo de alto nível que representa as principais entidades (objetos) e seus relacionamentos.

As principais entidades seriam:

- **Missão: Representa uma expedição espacial específica.**
- **Tripulação: Os membros envolvidos na missão.**
- **Carga Útil: Descrição dos equipamentos científicos transportados pela missão.**

Os relacionamentos incluem:

- **Uma Missão pode ter vários membros de Tripulação.**
- **Uma Missão pode ter uma ou mais Cargas Úteis.**

2.2 Modelagem Lógica:

Na modelagem lógica, traduzimos o modelo conceitual em estruturas de dados específicas para o banco de dados. Criaremos tabelas no banco de dados SQLite para representar nossas entidades:

Tabela "Missão":

- **ID da Missão: Chave primária (identificador único).**
- **Nome da Missão: Texto (string).**
- **Data de Lançamento: Data.**
- **Destino: Texto (string).**
- **Estado da Missão: Texto (string).**
- **Duração da Missão: Intervalo de tempo (duração).**
- **Custo da Missão: Moeda (decimal).**
- **Status da Missão: Texto longo (string ou texto).**

Tabela "Tripulação":

- **ID da Tripulação: Chave primária.**
- **ID da Missão: Chave estrangeira (relacionamento com a tabela "Missão").**
- **Nome do Membro: Texto (string).**

Tabela "Carga Útil":

- **ID da Carga Útil: Chave primária.**
- **ID da Missão: Chave estrangeira.**
- **Descrição: Texto (string).**

2.3 Normalização:

A normalização é importante para evitar redundância e inconsistências nos dados. Garantiremos que nossas tabelas estejam na Terceira Forma Normal (3NF), minimizando dependências funcionais e mantendo a integridade dos dados.

2.4 Indexação:

Criaremos índices nas colunas relevantes para otimizar as consultas. Por exemplo, podemos indexar o ID da Missão para facilitar a recuperação rápida de informações.

2.5 Considerações de Desempenho:

Avaliaremos o desempenho do banco de dados, considerando o volume de dados e as operações frequentes. Otimizaremos consultas e transações para garantir que o sistema seja eficiente.

Fase 3: 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. Aqui estão os detalhes:

3.1 Rotas da Aplicação (Flask):

a. Visualizar Todas as Missões

- **Criaremos uma rota que exibirá uma lista de todas as missões registradas no sistema.**
- **Os detalhes básicos de cada missão, como nome, destino e estado, serão mostrados.**
- **As missões serão ordenadas em ordem decrescente de data de lançamento.**

b. Recuperar Detalhes de uma Missão Específica

- **Criaremos uma rota para recuperar os detalhes de uma missão com base no ID da missão.**
- **Os usuários poderão visualizar informações detalhadas sobre uma missão específica.**

c. Pesquisar Missões por Intervalo de Datas

- **Implementaremos uma funcionalidade que permitirá aos usuários pesquisar missões com base em um intervalo de datas específico.**
- **Por exemplo, os usuários poderão ver todas as missões lançadas entre uma data inicial e uma data final.**

d. Adicionar Nova Missão

- **Criaremos um formulário para adicionar novas missões ao sistema.**
- **Os usuários poderão preencher campos como nome da missão, data de lançamento, destino, estado, duração, custo e status.**

e. Editar Missão Existente

- **Implementaremos um formulário de edição para modificar os detalhes de uma missão existente.**

- **Os usuários poderão atualizar informações como destino, tripulação, carga útil, etc.**

f. Excluir Missão

- **Criaremos uma opção para excluir uma missão do sistema com base no ID da missão.**
- **Isso removerá todos os dados associados à missão.**

3.2 Integração com o Banco de Dados SQLite:

- **Conectaremos nossa aplicação Flask ao banco de dados SQLite.**
- **As rotas mencionadas acima interagirão diretamente com as tabelas "Missão", "Tripulação" e "Carga Útil".**

3.3 Páginas HTML:

- **Criaremos páginas HTML para interagir com o usuário.**
- **Essas páginas exibirão os formulários, listas de missões e detalhes específicos.**

3.4 Testes:

- **Realizaremos testes para garantir que todas as funcionalidades estejam funcionando corretamente.**
- **Verificaremos se as operações CRUD estão sendo executadas conforme o esperado.**

Fase 4: 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:

4.1 Configuração do Ambiente:

- **Instalaremos o Python, o Flask.**
- **Configuraremos o ambiente de desenvolvimento.**

4.2 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**

4.3 Criação das Rotas:

- **No arquivo routes.py, criaremos as rotas para cada funcionalidade:**
  - **Visualizar todas as missões.**
  - **Recuperar detalhes de uma missão específica.**
  - **Adicionar nova missão.**
  - **Editar missão existente.**
  - **Excluir missão.**

4.4 Integração com o Banco de Dados SQLite:

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

4.5 Páginas HTML:

- **Criaremos páginas HTML na pasta templates para exibir as informações:**
  - **index.html: Lista de todas as missões.**
  - **mission_details.html: Detalhes de uma missão específica.**
  - **edit_mission.html: Formulário de edição para modificar os detalhes de uma missão.**

4.6 Estilização:

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

4.7 Testes:

- **Realizaremos testes para garantir que todas as funcionalidades estejam funcionando corretamente.**
- **Verificaremos se as operações CRUD estão sendo executadas conforme o esperado.**

4.8 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.

Lições Aprendidas: Ao longo do projeto, aprendemos a importância da comunicação transparente e da flexibilidade. A capacidade de se adaptar rapidamente a mudanças e de ouvir atentamente os feedbacks foi fundamental para o nosso sucesso. Além disso, a colaboração entre a equipe e a aplicação de metodologias ágeis nos permitiram alcançar resultados.

Próximos Passos: Para manter o ímpeto, recomendamos a realização de uma análise pós-implementação detalhada. Isso nos ajudará a identificar oportunidades de otimização contínua e a garantir que o projeto continue a evoluir de acordo com as necessidades.

Em resumo, este projeto não é apenas um sucesso isolado, mas também um exemplo inspirador de como podemos enfrentar desafios com criatividade, colaboração, paciência e determinação.
