1. Visão Geral

O sistema foi desenvolvido com o propósito de uma solução eficiente para o gerenciamento de tarefas, para a atribuição de usuários responsáveis por executá-las. A principal funcionalidade consiste em permitir o registro e o acompanhamento das horas que cada usuário dedica às atividades. Este sistema se destina especialmente a ambientes colaborativos, como equipes de desenvolvimento, grupos de trabalho em empresas ou times de projeto, onde há a necessidade de planejar, organizar, distribuir e monitorar atividades de maneira estruturada. A utilização do sistema facilita o controle de progresso de tarefas, a comunicação entre os membros da equipe e a transparência na alocação de recursos ao longo de projetos em andamento.

2. Decisões Arquiteturais

Foi adotada uma arquitetura baseada no Django REST Framework, aproveitando o padrão de desenvolvimento MVC (Model-View-Controller), que no contexto de APIs REST assume a forma Model-View-Serializer. Essa abordagem arquitetural foi escolhida por oferecer diversos benefícios, como a solidez e maturidade do framework Django, além da sua excelente integração com bancos de dados relacionais, com destaque para o MySQL. Outro fator foi o suporte do Django a funcionalidades essenciais como autenticação de usuários, controle de permissões de acesso, e serialização de dados entre modelos Python e formatos como JSON. A efetivação do padrão REST foi fundamental para garantir a interrupção da aplicação, viabilizando futuras integrações com outros sistemas e serviços web, como front-ends baseados em JavaScript ou sistemas corporativos.

3. Modelagem de Dados

A estrutura de dados do sistema foi planejada segundo o modelo relacional, utilizando tabelas com chaves primárias e estrangeiras para representar as relações entre os elementos. As principais entidades do banco de dados são:

-User: Representa os usuários que interagem com o sistema, armazenando dados como identificador único, nome de usuário, endereço de e-mail, senha, entre outros atributos.

-Task: Refere-se as tarefas criadas e gerenciadas na plataforma. Cada tarefa possui campos como título, descrição, status de andamento e data de criação.

-UserTask: Esta é uma tabela de associação que estabelece a relação entre os usuários e as tarefas, incluindo um campo específico para o registro do número de horas que determinado usuário dedicou a uma determinada tarefa.

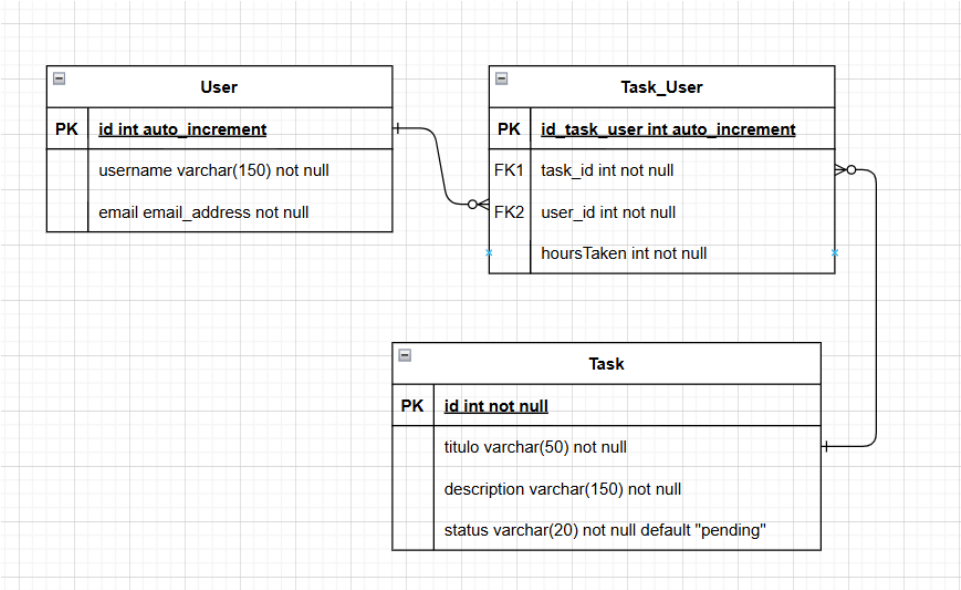
Descrição das tabelas:

- User: id, username, email, password, etc.

- Task: id, title, description, status, created_at, etc.

- UserTask: id, user_id, task_id, hours_spent

Um diagrama entidade-relacionamento pode ser utilizado para ilustrar graficamente essas conexões, facilitando o entendimento da estrutura do banco.



4. Fluxo de Requisições

Principais endpoints da API:
- GET /users/ – Lista usuários
- POST /users/ – Cria novo usuário
- GET /tasks/ – Lista tarefas
- POST /tasks/ – Cria nova tarefa
- GET /user-tasks/ – Lista atribuições de usuários a tarefas
- POST /user-tasks/ – Atribui usuário a tarefa

Exemplo de uso para criar uma tarefa:

```
POST /tasks/
{
  "title": "Documentar projeto",
```

```
  "description": "Criar documentação técnica",
  "status": "Pendente"
}
```

5. Configuração e Deploy

Dependências:
- Python 3.x
- Django
- Django REST Framework
- MySQL
- Outros pacotes listados em requirements.txt

Passos para execução:

1. Instale as dependências:
   pip install -r requirements.txt
2. Configure o banco de dados em settings.py.
3. Aplique as migrações:
   python manage.py migrate
4. Crie um superusuário (opcional):
   python manage.py createsuperuser
5. Inicie o servidor:
   python manage.py runserver