# 1. Visão Geral

O sistema tem como objetivo gerenciar tarefas e a atribuição de usuários a essas tarefas, permitindo o controle de horas dedicadas por cada usuário. O contexto de uso envolve equipes de trabalho que precisam organizar, distribuir e acompanhar o progresso de atividades em projetos, facilitando a gestão e a colaboração.

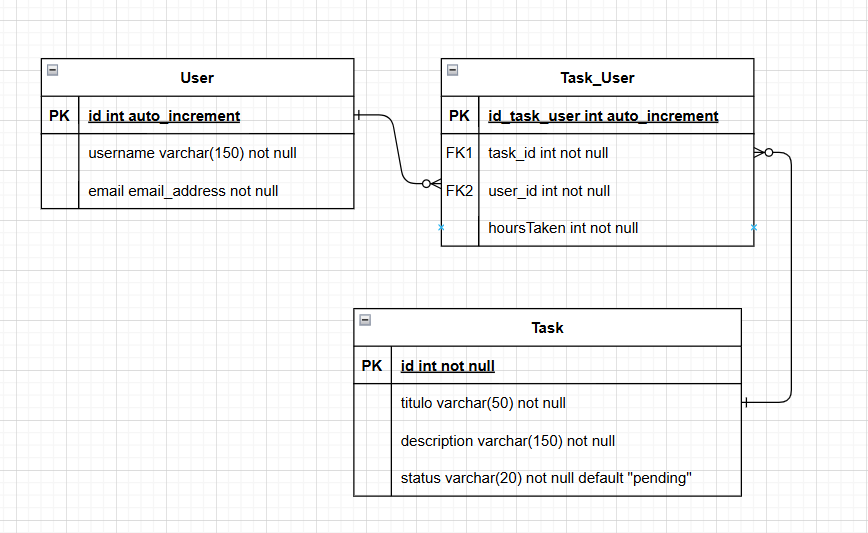
# 2. Decisões Arquiteturais

Optou-se pela arquitetura baseada em Django REST Framework, utilizando o padrão MVC (Model-View-Controller) adaptado para web APIs (Model-View-Serializer). Essa escolha se justifica pela robustez do Django, facilidade de integração com bancos relacionais (MySQL) e suporte nativo a autenticação, permissões e serialização de dados. O padrão REST foi adotado para garantir interoperabilidade e facilitar integrações futuras.

# 3. Modelagem de Dados

O banco de dados segue o modelo relacional, com as seguintes tabelas principais:  
- User: Armazena informações dos usuários do sistema.  
- Task: Representa as tarefas cadastradas.  
- UserTask: Tabela de associação entre usuários e tarefas, incluindo o registro de horas dedicadas.

O diagrama entidade-relacionamento está ilustrado abaixo:



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

# 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