МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Программная инженерия»

Проектный практикум

Kanban

Отчёт

Выполнил: Студент группы

БПИ2403

Рыбаченок Вадим

Москва

2025

Проектирование Kanban

Цель: Разработка бэкенда для Kanban-доски с поддержкой:

- CRUD-операций для проектов, колонок и задач.
- Мягкого удаления через флаг `is_active`.
- Авторизации через JWT.
- Работы с базой данных (SQL Server) и ORM (SQLAlchemy).

Зависимости:

```
fastapi>=0.68.0,<0.70.0
uvicorn>=0.15.0
sqlalchemy>=1.4.0,<2.0.0
pydantic>=1.8.0,<2.0.0
python-jose>=3.3.0
passlib>=1.7.0
pyodbc>=4.0.0
python-multipart>=0.0.5
anyio>=3.0.0
```

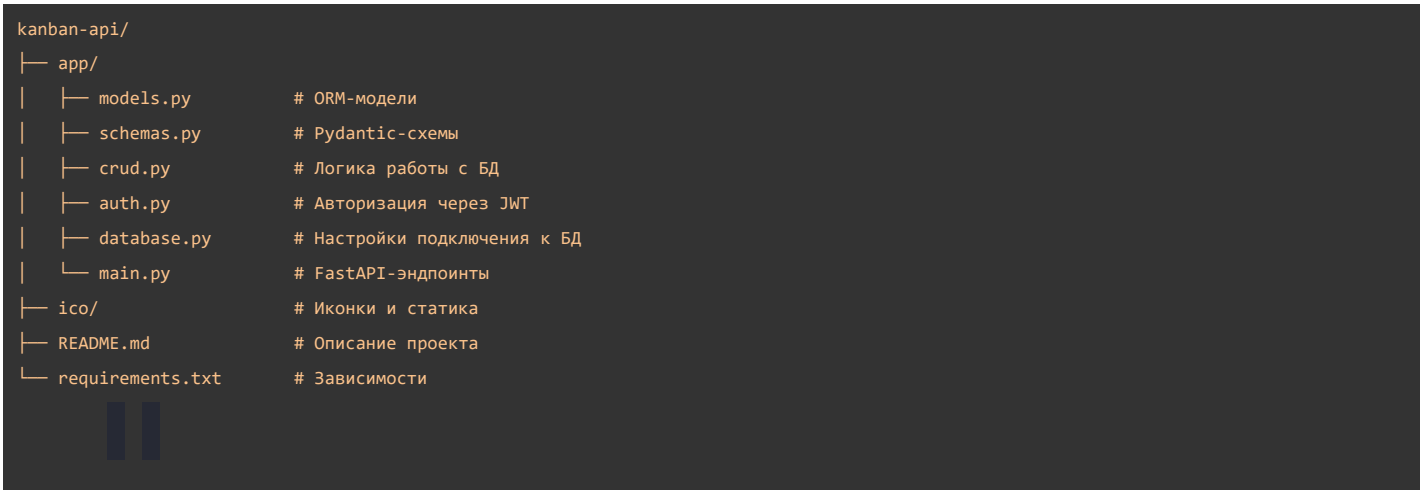
Рисунок №1 – Зависимости

Дерево зависимостей:

```
fastapi
├── uvicorn
├── pydantic
├── sqlalchemy
├── python-jose
└── passlib

sqlalchemy
└── pyodbc
```

Рисунок №2 – Структура проекта



Дизайн API

Архитектурный подход:
REST API : Все операции выполняются через HTTP-методы (GET/POST/PUT/DELETE).
JWT-токены : Авторизация через Bearer-токены.
Мягкое удаление : Проекты, колонки и задачи деактивируются через поле is_active.

Пример инициализации экземпляра API

```
from fastapi import FastAPI
from .database import engine
from . import models

app = FastAPI()
models.Base.metadata.create_all(bind=engine)
```

Основные функции

1.Авторизация и безопасность

- JWT-токены: Генерация через create_access_token
- Хеширование паролей: Использование bcrypt через passlib

2.CRUD-операции

ФУНКЦИЯ	МЕТОД	ЭНДПОИНТ	ОПИСАНИЕ
Создание пользователя	POST	`/users/`	Регистрация нового пользователя
Создание проекта	POST	`/projects/`	Доступно только авторизованным
Получение задач по колонке	GET	`/columns/{column_id}/tasks/`	С фильтрацией по приоритету
Мягкое удаление проекта	DELETE	`/projects/{project_id}`	Деактивация через `is_active=False`
Восстановление проекта	POST	`/projects/{project_id}/restore`	Установка `is_active=True`

3. Мягкое удаление через is_active

- Реализация:
    - Project.is_active: Mapped[bool] = mapped_column(Boolean, default=True)
    - Column.is_active, Task.is_active
- Логика:
    - При удалении объекта is_active устанавливается false
    - Все запросы фильтруют активные объекты

```
db.query(models.Project).filter(models.Project.is_active == True)
```

4. Работа с колонками и задачами

- Создание задачи:
    - Проверка активности колонки

```
column = db.query(models.Column).filter(
    models.Column.id == column_id,
    models.Column.is_active == True
).first()
```

    - Пример эндпоинта:

```
@app.post("/columns/{column_id}/tasks/",
response_model=schemas.TaskResponse)
def create_task(...):
```

Примеры использования

1. Регистрация пользователя

```
POST /users/
{
  "email": "admin@example.com",
  "password": "secret"
}
```

2. Создание проекта

```
POST /projects/
Authorization: Bearer <token>
{
  "name": "Проект А"
}
```

3. Получение задач по колонке

```
GET /columns/1/tasks/?priority=2
{
  "id": 1,
  "title": "Задача 1",
  "priority": 2
}
```

Реализация ключевых компонентов

1. Модель Project

```
class Project(Base):
    __tablename__ = "projects"
    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    name: Mapped[str] = mapped_column(String(255))
    owner_id: Mapped[int] =
mapped_column(ForeignKey("users.id"))
    is_active: Mapped[bool] = mapped_column(Boolean,
default=True)
    members: Mapped[list["User"]] =
relationship(secondary="project_members",
back_populates="projects")
    columns: Mapped[list["Column"]] =
relationship(back_populates="project")
```

2. Мягкое удаление в crud.py

```
def delete_project(db: Session, project_id: int):
    project = get_project(db, project_id)
    if not project:
        raise HTTPException(status_code=404, detail="Проект не
найден")
    project.is_active = False
    db.commit()
    return {"message": "Проект деактивирован"}
```

3. Получение проектов текущего пользователя

```
@app.get("/projects/me/",
response_model=list[schemas.ProjectDetails])
def read_user_projects(db: Session = Depends(get_db),
current_user: models.User = Depends(get_current_user)):
    # Логика фильтрации по `is_active`
```

Вывод

Бэкенд для Kanban-доски соответствует всем требованиям:

- Реализованы CRUD-операции с контролем прав доступа.
- Внедрено мягкое удаление через is_active.
- Авторизация через JWT.
- Проект готов к интеграции с фронтендом и расширению функционала (например, добавление ролей или веб-хуков для уведомлений).

Репозиторий на GITHUB https://github.com/redlyaguha/kanban