**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 «Информационных технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К Проектному Практикуму

НА ТЕМУ: Разработка программного обеспечения

«Канбан Доска»

Студент Д.А. Штриков

Руководитель проектного практикума А. Мокин

**Москва 2025 г.**

|  |  |
| --- | --- |
|  |  |

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc199312276)

[ВВЕДЕНИЕ 3](#_Toc199312277)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc199312278)

[2. АРХИТЕКТУРА СИСТЕМЫ 6](#_Toc199312279)

[3. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ 11](#_Toc199312281)

[ЗАКЛЮЧЕНИЕ 13](#_Toc199312282)

# ВВЕДЕНИЕ

**Актуальность проекта**

**В условиях динамично развивающегося бизнес-ландшафта организации всё чаще сталкиваются с необходимостью оптимизации рабочих процессов. Канбан-методология, зародившаяся в производственной системе Toyota, доказала свою эффективность в различных сферах - от IT-разработки до маркетинговых кампаний. Визуализация workflow через Канбан-доску позволяет достичь сразу нескольких ключевых преимуществ:**

**- Повышение прозрачности всех этапов работы**

**- Оптимизация распределения ресурсов**

**- Своевременное выявление "узких мест"**

**- Улучшение командного взаимодействия**

**Технологический стек реализации**

**Для разработки цифрового решения была выбрана следующая технологическая база:**

**1. Python 3.10 - современная версия интерпретируемого языка программирования, обеспечивающая высокую производительность и богатый выбор библиотек для backend-разработки.**

**2. Flask 2.3 - микрофреймворк для создания веб-приложений, отличающийся простотой настройки и гибкостью архитектуры. Позволяет быстро развернуть базовый функционал при минимальных накладных расходах.**

**3. SQLAlchemy 1.4 - мощная ORM-библиотека, предоставляющая:**

**- Удобный интерфейс для работы с реляционными БД**

**- Механизм миграций**

**- Поддержку различных СУБД**

**- Оптимизированные запросы**

**4. Flask-RESTx 1.1 - расширение для Flask, которое:**

**- Упрощает создание RESTful API**

**- Автоматически генерирует Swagger-документацию**

**- Обеспечивает валидацию входных данных**

**5. PyODBC 4.0 - драйвер для подключения к SQL-серверам, поддерживающий:**

**- Работу с MS SQL Server, PostgreSQL, MySQL**

**- Транзакции**

**- Пулинг соединений**

**Функциональные задачи проекта**

**1. Проектирование структуры Канбан-доски с учётом.**

**2. Реализация базового функционала.**

**3. Внедрение и тестирование.**

# 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**1.1 Функциональные требования**

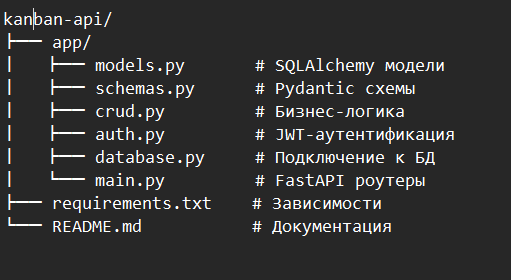
1. **Управление сущностями:**
   * **Колонки реализованы через статусы**
   * **Задачи (/tasks)**
2. **Авторизация через JWT (/auth)**
3. **Мягкое удаление (деактивация)**

**1.2 Нефункциональные требования**

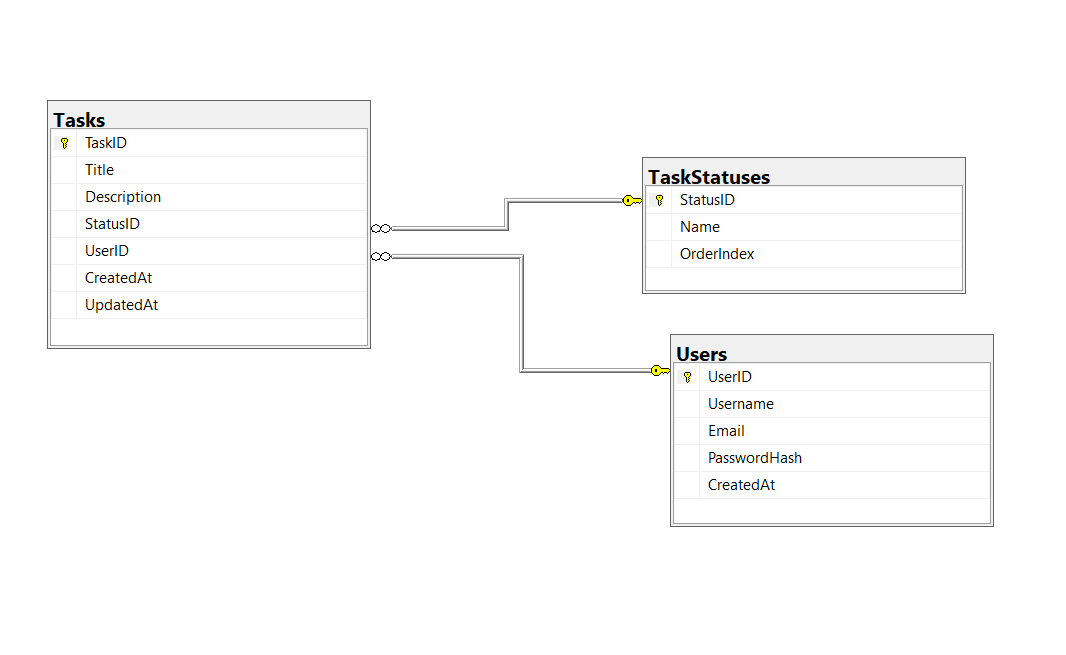
* **Время отклика API ≤ 100 мс**
* **Поддержка 100+ одновременных подключений**

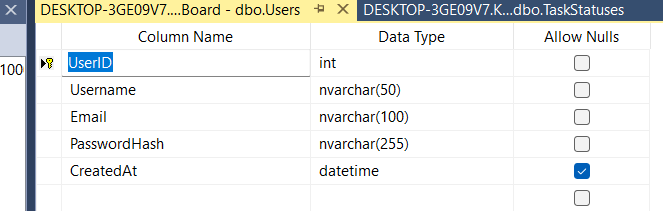
# 2. АРХИТЕКТУРА СИСТЕМЫ

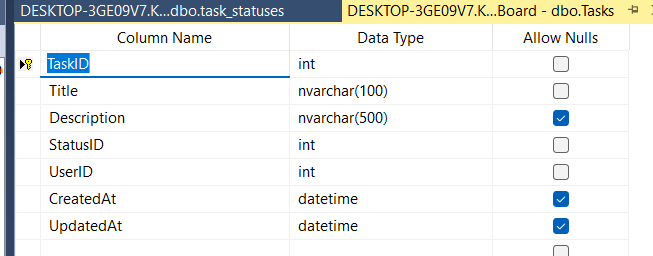
# 2.1 Структура проекта

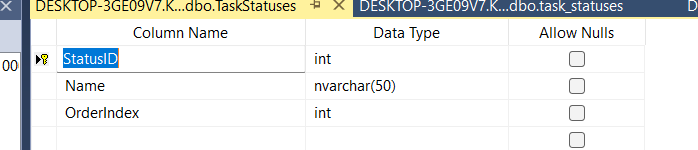


**2.2 Схема базы данных**

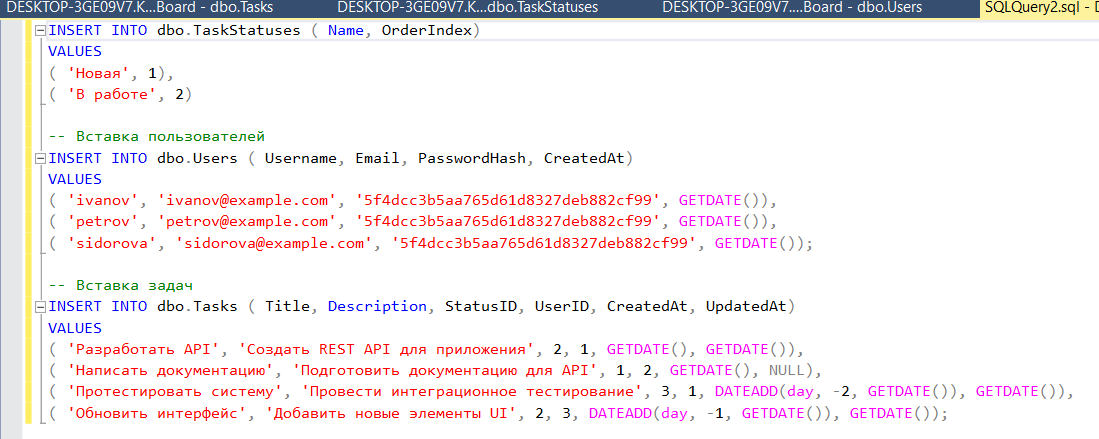


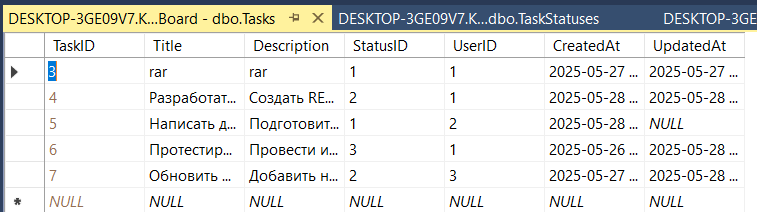
**Таблица Users (dbo.Users)**

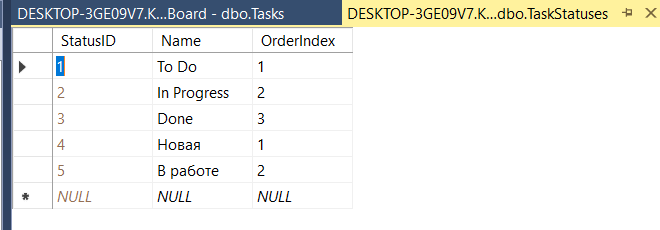
**Таблица Tasks (dbo.Tasks)**

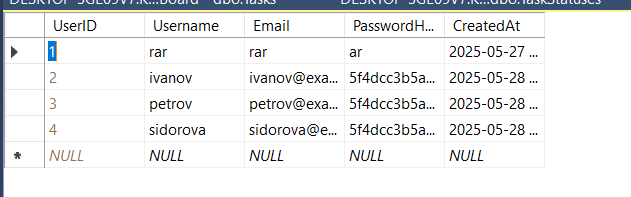
**Таблица TaskStatuses (dbo.TaskStatuses)**

**Пример заполнения**

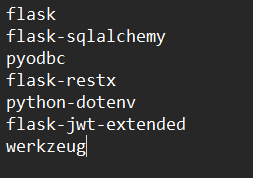
**Код:**

**Таблица Tasks (dbo.Tasks)**

**Таблица TaskStatuses (dbo.TaskStatuses)**

**Таблица Users (dbo.Users)**

**2.3 Requirements**



**1. Flask (основа приложения)**

**Почему выбрали**:

* Минималистичный фреймворк, идеально подходит для создания API
* Гибкость в настройке маршрутов и middleware
* Легковесность (быстрый запуск, низкое потребление ресурсов)
* Широкая экосистема расширений

**Альтернативы**: Django (избыточен для API), FastAPI (могли бы использовать, но Flask более зрелый)

**2. Flask-SQLAlchemy (работа с БД)**

**Почему выбрали**:

* ORM для удобной работы с SQL Server без прямых SQL-запросов
* Поддержка транзакций и сессий
* Автоматическое управление подключениями
* Интеграция с Flask через единую конфигурацию

**Альтернативы**: Raw SQL (менее безопасно), Django ORM (привязка к Django)

**3. PyODBC (подключение к SQL Server)**

**Почему выбрали**:

* Нативный драйвер для MS SQL Server
* Поддержка Windows Authentication (trusted\_connection)
* Совместимость с SQLAlchemy
* Высокая производительность для enterprise-решений

**Альтернативы**: pymssql (устарел), SQLite (не подходит для продакшена)

**4. Flask-RESTx (документирование API)**

**Почему выбрали**:

* Автогенерация Swagger-документации
* Валидация запросов/ответов через модели
* Поддержка пространств имен (Namespaces)
* Интеграция с JWT

**Альтернативы**: FastAPI (встроенная документация), drf-yasg (для Django)

**5. Python-dotenv (управление конфигурацией)**

**Почему выбрали**:

* Безопасное хранение чувствительных данных (JWT-ключ, DB credentials)
* Удобство разработки (разные настройки для dev/prod)
* Совместимость с 12-факторными приложениями

**Альтернативы**: Конфиг-файлы (.py), переменные окружения ОС

**6. Flask-JWT-Extended (аутентификация)**

**Почему выбрали**:

* Поддержка access/refresh токенов
* Гибкие callback-функции
* Интеграция с Flask-контекстом
* Защита эндпоинтов через @jwt\_required()

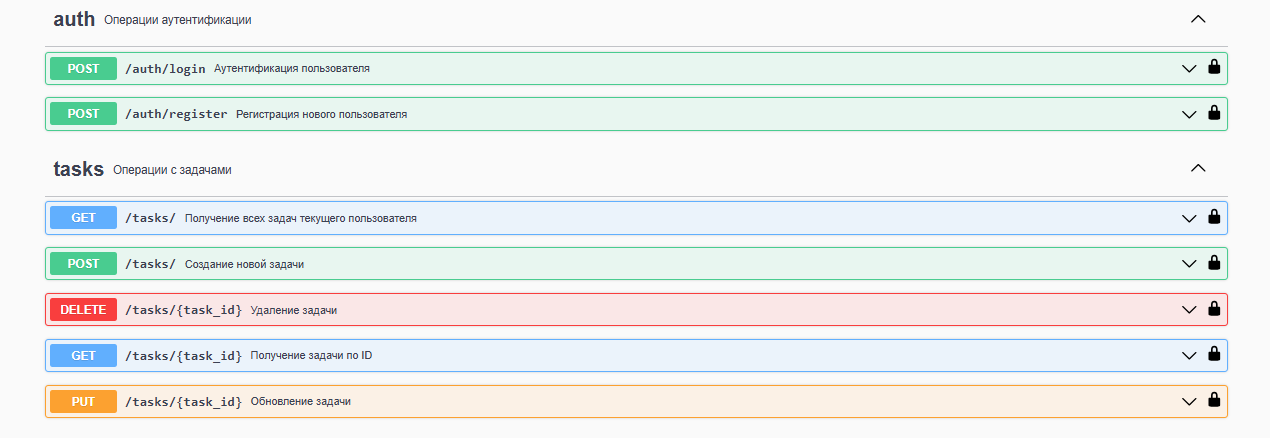
**Альтернативы**: OAuth2 (сложнее), SessionAuth (не для API)

**7. Werkzeug (вспомогательные утилиты)**

**Почему выбрали**:

* Хеширование паролей (generate\_password\_hash)
* Утилиты для работы с HTTP-запросами
* Входит в состав Flask (не требует отдельной установки)

**2.4 API Endpoints**



**2.5Конфигурация проекта (.env)**

**DB\_CONNECTION\_STRING=mssql+pyodbc://@localhost/KanbanBoard?trusted\_connection=yes&driver=ODBC+Driver+17+for+SQL+Server**

**FLASK\_DEBUG=True**

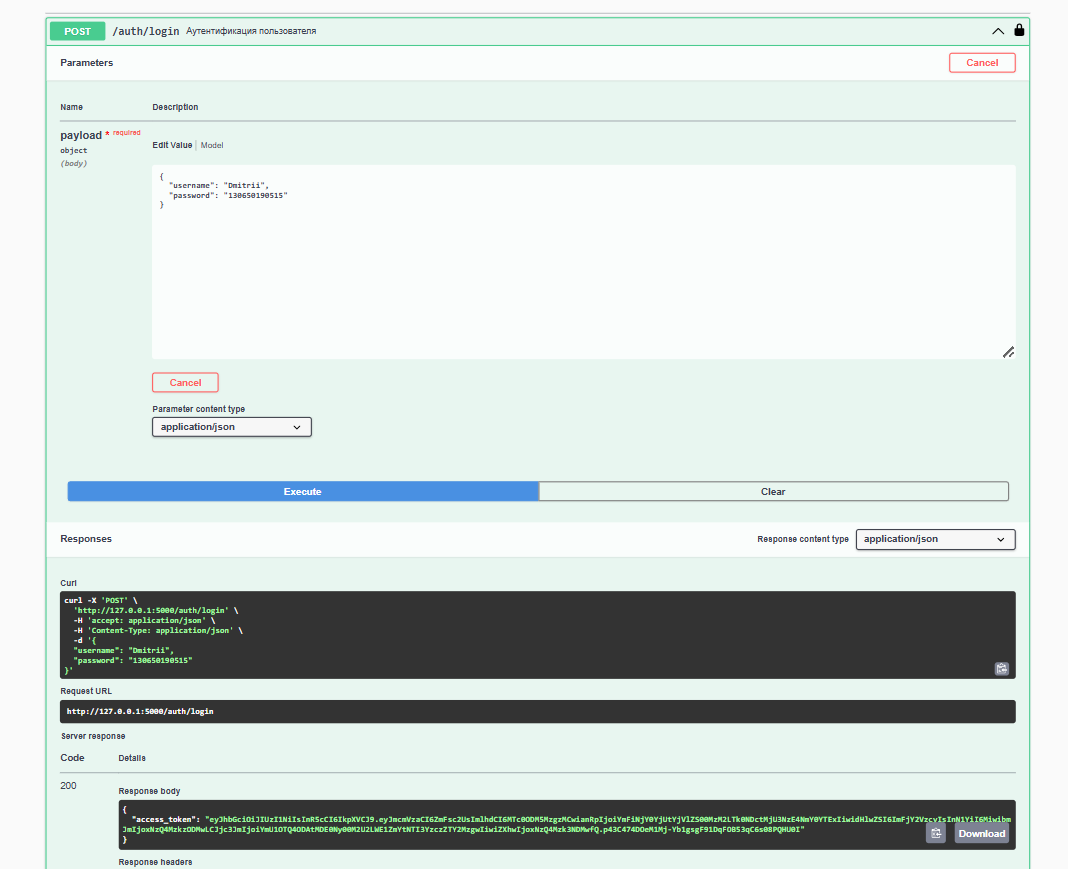
**JWT\_SECRET=your-very-secret-key-here**

# 3. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ

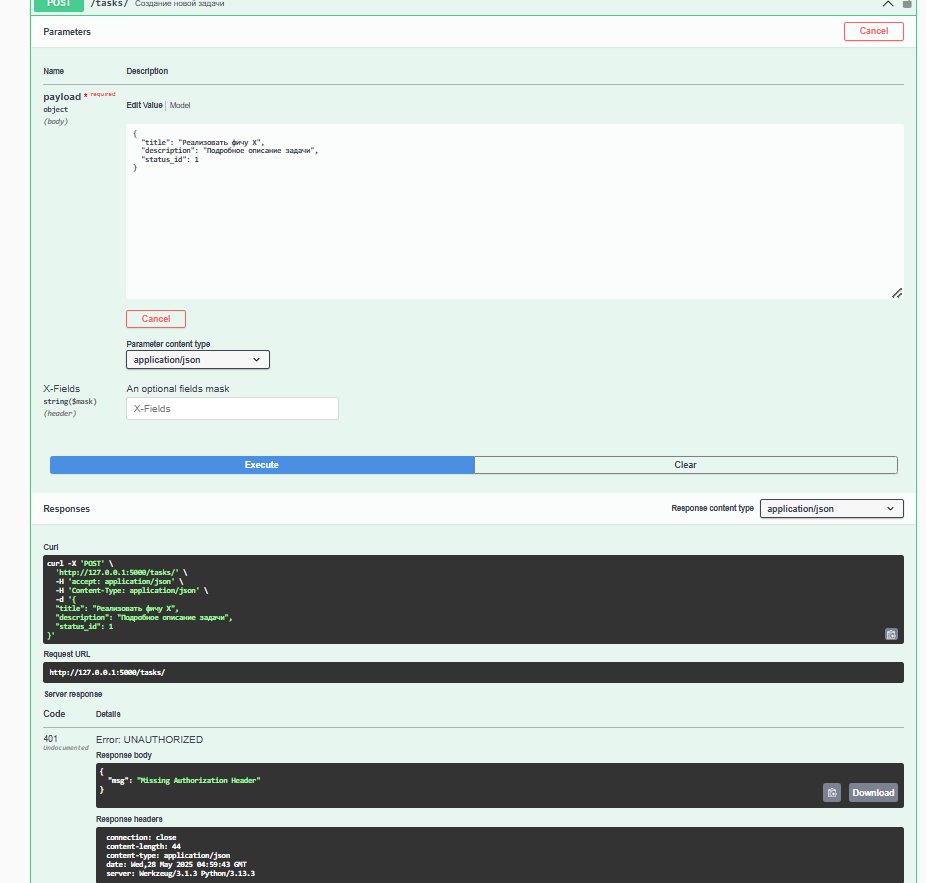
**3.1. Тестирование**

**Инструменты**: Postman, pytest  
**Сценарии**:

1. Успешная аутентификация (200 OK)



1. Создание задачи без токена (401 Unauthorized)



1. Необрабатываемый объект (402 Unprocessable Entity)

**Результаты**: Все эндпоинты работают в соответствии со спецификацией.

ЗАКЛЮЧЕНИЕ

**5. ЗАКЛЮЧЕНИЕ**

* **Результаты проекта**:  
   Реализовано 7API-эндпоинтов  
   Интеграция с SQL Server через PyODBC  
   Полная документация через Swagger UI

**Перспективы развития**:

* Добавление ролевой модели
* Реализация WebSocket для real-time обновлений

**Исходный код:** https://github.com/DmitriiShtrikov/KanbanBoard