**Финальная проверка: выполнены ли все требования ТЗ**

**Я сверил изначальные требования с тем, что уже реализовано.   
Ниже — полный отчёт по Заданию 42.6.1 (HW-04).**

**📌 1. Базовые функции API**

| **Требование** | **Статус** |
| --- | --- |
| **Создание код-сниппетов** | ✅ Реализовано (POST /snippets/) |
| **Редактирование код-сниппетов** | ✅ Реализовано (PUT /snippets/{id}) |
| **Удаление код-сниппетов** | ✅ Реализовано (DELETE /snippets/{id}) |
| **Хранение сниппетов на сервере** | ✅ Реализовано (SQLite + SQLAlchemy ORM) |
| **Генерация ссылки для доступа к сниппету** | ✅ Реализовано (поиск по uuid) |
| **Управление доступом (авторизация, права доступа)** | ✅ Реализовано (JWT + get\_current\_user()) |
| **Поиск и фильтрация сниппетов** | ✅ Реализовано (GET /snippets?language=Python) |

**Все базовые функции API выполнены.**

**2. Архитектура приложения**

| **Требование** | **Статус** |
| --- | --- |
| **Python + FastAPI + SQLAlchemy** | ✅ Используем FastAPI + Async SQLAlchemy |
| **Структура проекта по "чистой архитектуре"** | ✅ Разделение на core, api, models, schemas, services, db, tests |
| **Асинхронное подключение к базе данных** | ✅ Используется AsyncSession с SQLAlchemy |

**Структура соответствует "чистой архитектуре".**

**3. Аутентификация и авторизация (JWT)**

| **Требование** | **Статус** |
| --- | --- |
| **Использование JWT для авторизации** | ✅ Реализовано (access\_token, refresh\_token) |
| **Использование python-jose для работы с JWT** | ✅ Реализовано |
| **Использование passlib[bcrypt] для хеширования паролей** | ✅ Реализовано |
| **Использование bcrypt для безопасного хранения паролей** | ✅ Реализовано |

**Аутентификация и защита API работают корректно.**

**4. Работа с код-сниппетами**

| **Требование** | **Статус** |
| --- | --- |
| **Авторизованный пользователь может создавать, редактировать и удалять код-сниппеты** | ✅ Реализовано |
| **Другие пользователи могут запрашивать сниппет по uuid без авторизации** | ✅ Реализовано |
| **Данные пользователей и код-сниппеты хранятся в базе данных** | ✅ Реализовано |

**Доступ к сниппетам и управление правами реализованы.**

**5. Оптимизация и производительность**

| **Требование** | **Статус** |
| --- | --- |
| **Асинхронная работа с БД** | ✅ Используется AsyncSession |
| **Использование индексов для ускорения поиска** | ✅ Реализовано (language, owner\_id) |
| **Фильтрация сниппетов по language и owner\_id** | ✅ Реализовано (GET /snippets?language=Python) |

**Оптимизация базы данных выполнена.**

**6. Логирование и обработка ошибок**

| **Требование** | **Статус** |
| --- | --- |
| **Обработка ошибок через middleware** | ✅ Реализовано (exception\_handler) |
| **Обработка ошибок через exception\_handler** | ✅ Реализовано |
| **Логирование всех действий (регистрация, вход, изменение сниппетов)** | ✅ Реализовано (logs/api.log) |
| **Структурированные логи** | ✅ Реализовано (logging.Logger) |

**Логирование и обработка ошибок работают корректно.**

**7. WebSockets и уведомления**

| **Требование** | **Статус** |
| --- | --- |
| **WebSockets-уведомления при изменении сниппетов** | ✅ Реализовано |
| **WebSocket-клиент (ws\_client.py) получает уведомления** | ✅ Реализовано |
| **Поддержка уведомлений о редактировании и удалении** | ✅ Реализовано |

**WebSockets-уведомления работают.**

**Все требования полностью выполнены.** ✅

**Какие тесты прошли успешно:**

1. **Регистрация, логин, обновление токена.**
2. **Создание, редактирование, удаление сниппетов.**
3. **Фильтрация и поиск сниппетов (GET /snippets?language=Python).**
4. **WebSockets-уведомления при изменении сниппетов.**

**Вывод**: все задачи **выполнены**, тесты **пройдены успешно**, проект завершён!