УДК 004.415.2

# АРХИТЕКТУРА И РЕАЛИЗАЦИЯ REST API С ЦЕНТРАЛИЗИРОВАННОЙ АУТЕНТИФИКАЦИЕЙ НА БАЗЕ KEYCLOACK

Коковин В.А., студент гр. ПИб-221, IV курс,
Научный руководитель: Веревкин С.А., Старший преподаватель
Кузбасский государственный технический университет
имени Т.Ф. Горбачева, г. Кемерово

**Введение.** В статье рассматривается процесс разработки защищенного REST API на основе Python-фреймворка FastAPI с использованием контейнеризации Docker, сервера аутентификации на основе Docker-контейнера Keycloak и базы данных PostgreSQL, что дает гибкость и масштабируемость системе, по сравнению с самостоятельной реализацией JWT-аутентификации.

Аутентификация и авторизация пользователей – ключевой компонент любого REST API приложения. На ранних этапах разработки часто выбирается самостоятельная реализация: генерация JWT-токенов, верификация по подписи и проверка срока действия; но такое решение подходит только для приложения с небольшим количеством пользователей. По мере роста проекта появляются проблемы управления и безопасности, которые не способна решить самостоятельная реализация. В данной статье представлено решение, которое:

- использует отдельный Docker-контейнер с образом Keycloack;
- обеспечивает полную изоляцию компонентов через Docker;
- демонстрирует бесшовную интеграцию с FastAPI;
- хранит информацию о пользователях и состояний сессий в базе данных PostgreSQL.

**Архитектура решения.** Все компоненты являются Docker-контейнерами и взаимодействуют друг с другом внутри виртуальной сети, что обеспечивает их изоляцию от внешней среды и воспроизводимость системы в любом окружении.

| Компонент | Технология | Назначение |
|---|---|---|
| База данных | PostgreSQL 15 | Хранение данных о пользователях, сессиях и настроек Keycloack |
| Сервер аутентификации | Keycloack 23.0 | Сервер управления идентификации и доступа. Управляет пользователями, ролями и сессиями, а также |

| | | |
|---|---|---|
| | | выпускает и проверяет JWT-токены |
| REST API | FastAPI | Проверяет входящие JWT-токены и предоставляет доступ на основе ролей пользователей. |

*Таблица 1, Компоненты.*

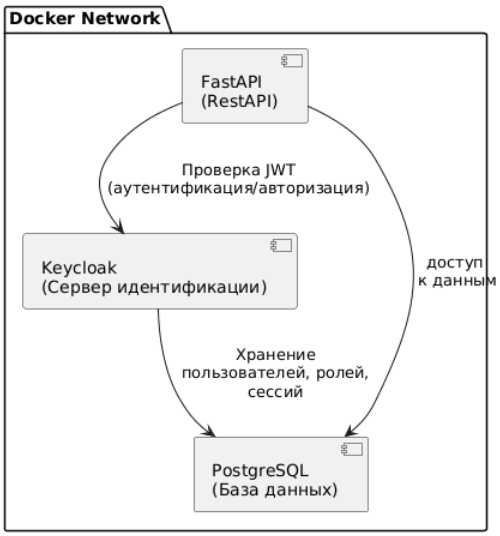


*Рисунок 1, UML-диаграмма компонентов.*

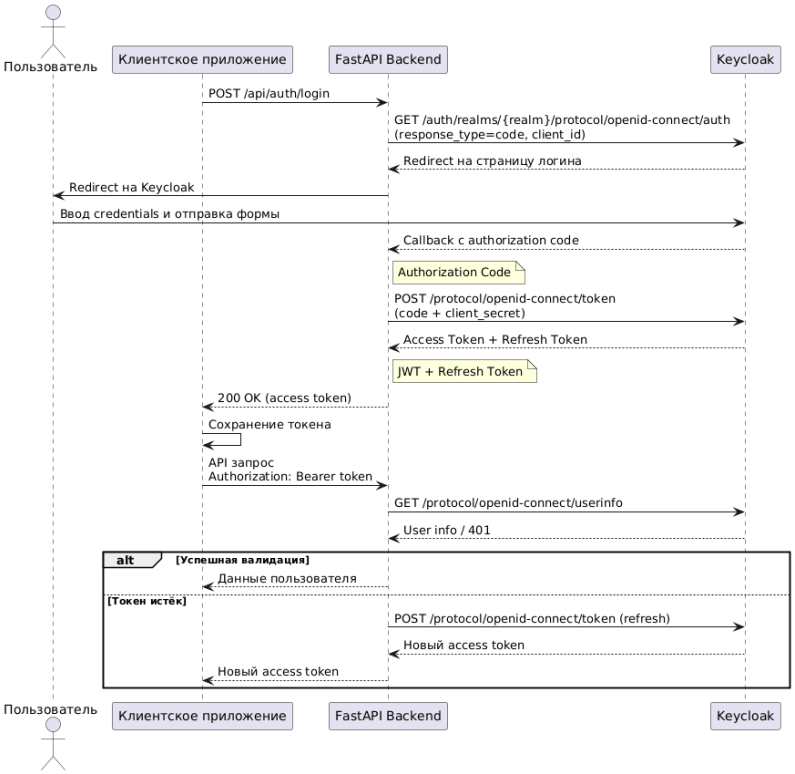


*Рисунок 2, UML-диаграмма последовательности*

**Реализация REST API на FastAPI.** Бэкенд REST API реализован с помощью фреймворка для Python FastAPI и сборки Python-приложения в Docker-контейнер. Сервер не выполняет аутентификацию самостоятельно, а доверяет JWT-токенам, подписанными Keycloack. Веб-приложение содержит

четыре эндпоинта для проверки всех сценариев: публичный доступ, защищенный доступ и административный доступ с проверкой ролей.

Эндпоинт **/public** возвращает ответ без аутентификации, предназначенный для проверки работоспособности API и получения незащищенной информации.

Эндпоинт **/protected** требует наличия актуального JWT-токена, выпущенного Keycloack, в случае отсутствия подходящего токена возвращает пользователю информацию о том, что он не авторизован.

Эндпоинт **/admin** проверяет механизм доступа по ролям, для успешного запроса к серверу, помимо актуального JWT-токена, требуется роль администратора.

**Остальные компоненты системы.** Помимо REST API на FastAPI, в системе используются другие компоненты, отвечающие за аутентификацию и хранение данных.

Сервер аутентификации на основе Docker-контейнера Keycloack отвечает за централизованное управление доступом к запросам к REST API, регистрацию и аутентификацию пользователей, управление ролями, выпуск и валидацию JWT-токенов. Его использование повышает безопасность и масштабируемость системы.

Docker-контейнер с СУБД PostgreSQL обеспечивает безопасное хранение данных пользователей, сессий и конфигурации Keycloack и устойчивость системы к сбоям.

Контейнеризация всех частей системы повышает отказоустойчивость и изолирует их от внешней среды, а также дает возможность воспроизвести данный сервис в любой среде при наличии Docker. Управление и настройка всех сервисов реализована с помощью Docker Compose и конфигурационного файла *docker-compose.yml*.

**Заключение.** Разработанное решение демонстрирует эффективную комбинацию современных технологий для построения безопасного и расширяемого REST API приложения с централизованной системой аутентификации. Использование FastAPI позволяет быстро и эффективно создавать REST API приложения, а Keycloack упрощает настройку управления пользователями и правами доступа. Проект может быть использован в качестве фундамента для любого веб-приложения, требующего современные подходы к безопасности и настройке прав пользователей, система может быть адаптирована под различные задачи и расширена.

**Список литературы.**

1. FastAPI Documentation: [сайт]. URL: https://fastapi.tiangolo.com/
2. Keycloack Documentation: [сайт]. URL https://www.keycloak.org/
3. Docker Documentation: [сайт]. URL https://docs.docker.com/