СИНХРОНИЗАЦИЯ ПРОФИЛЕЙ ПОЛЬЗОВАТЕЛЕЙ МЕЖДУ ПРОВАЙДЕРОМ KEYCLOAK IAM И РЕЛЯЦИОННОЙ БАЗОЙ ДАННЫХ В SPRING-ПРИЛОЖЕНИИ

Войтов А.В., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Нестеренков С.Н. – канд. техн. наук, доцент

В работе предложен метод JIT-синхронизации профилей между Keycloak и реляционной БД в Spring-приложении. Извлечение данных из JWT-токена «на лету» позволяет транзакционно создавать и обновлять локальные записи без прямых запросов к IAM, снижая сетевую нагрузку и сохраняя stateless-архитектуру.

В современных системах обучения аутентификацию и авторизацию всё чаще выносят в специализированные решения класса Identity and Access Management (IAM). В рамках разработки программного средства автоматизации процесса повышения квалификации сотрудников IT-компании роль такого IAM-провайдера выполняет Keycloak [1]. Серверная логика приложения при этом реализована на базе популярного фреймворка Spring Boot [2].

Keycloak надежно хранит учетные данные и поддерживает сквозную авторизацию (Single Sign-On). Однако из-за этого возникает классическая проблема распределенных систем: необходимость связывания идентификатора пользователя из Keycloak с его локальным профилем в реляционной базе PostgreSQL [3], где хранятся оценки и прогресс по курсам. Для корпоративной среды критически важно сохранять строгую целостность этих данных. Если сотрудник перейдет в другой отдел, история его обучения ни в коем случае не должна потеряться.

Существует возможность выполнения синхронных REST-запросов к Admin API Keycloak при каждом действии пользователя для получения профиля. Но это серьезно замедлит работу приложения и создаст слишком жесткую связь между сервисами. Хранить все оценки и прогресс прямо в атрибутах Keycloak также нецелесообразно: его структура совершенно не рассчитана на сложные реляционные связи и запросы.

Для эффективного решения этой проблемы был внедрен паттерн Just-In-Time (JIT) инициализации профилей. Данный механизм работает следующим образом: система перехватывает и разбирает токен доступа (JWT), передаваемый клиентским React-приложением в заголовках каждого запроса. Вся логика синхронизации встроена непосредственно в цепочку фильтров Spring Security. Для этого был разработан специальный компонент JwtAuthenticationConverter. Он обрабатывает входящий HTTP-запрос еще до попадания в слой контроллеров и собирает готовый объект авторизации для контекста безопасности. Параллельно конвертер преобразует роли пользователя: извлекается массив realm_access.roles из токена и преобразуется в список прав GrantedAuthority, чтобы аннотации безопасности на уровне сервисов работали корректно.

При успешной проверке подписи токена из него извлекаются базовые поля: уникальный UUID, почта, ФИО, а также дополнительные данные о корпоративной должности. Сразу после этого приложение обращается к локальной базе данных. Если записи с таким UUID еще нет, открывается новая транзакция, и в таблице создается свежий профиль сотрудника. Для предотвращения состояний гонки (race condition) и дублирования записей при одновременном поступлении нескольких запросов от нового пользователя, сохранение выполняется через строгую транзакцию, а колонка UUID в базе защищена ограничением уникальности. Если же профиль уже существует, осуществляется сверка его полей с данными из токена. В случае изменения ФИО или должности локальная база немедленно обновляется, что позволяет поддерживать согласованность данных в конечном счете (eventual consistency).

В итоге данный подход позволяет эффективно работать с бизнес-данными внутри монолита. Снижается сетевая нагрузка на Keycloak и исключается необходимость настройки очередей сообщений. Значительно повышается отказоустойчивость: даже при временном сбое сервера авторизации платформа продолжит обслуживать пользователей с действующими токенами. Важным преимуществом является сохранение архитектуры stateless (сервер не хранит локальные сессии в памяти). Следовательно, при увеличении числа пользователей обеспечивается простое и быстрое горизонтальное масштабирование узлов приложения.

Список использованных источников:

1. Keycloak Documentation [Электронный ресурс]. – Режим доступа: https://www.keycloak.org/documentation (дата обращения: 29.03.2026).

2. Walls, C. Spring in Action, Sixth Edition / C. Walls. – Manning, 2022. – 536 p.

3. PostgreSQL Documentation [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 29.03.2026).