LMS для очного формата обучения

Авторы: Венедиктов Р. М., Самойлов В. М., Туров К. В.

Ментор: Дурова Татьяна

Санкт-Петербургская школа физико-математических и компьютерных наук
НИУ ВШЭ - Санкт-Петербург

2021

Введение

У каждого курса для выставления оценок используются различные таблицы. У этого способа достаточно много недостатков:

- На их создание тратится много времени
- Их сложно агрегировать, кроме как ещё одной таблицей.
- Приходится постоянно заходить, чтобы узнать, не проверили ли работу месячной давности или не выложили ли новую
- Иногда до последнего момента не понятно, какая оценка выходит, если не потратить время на самостоятельное вычисление

Хочется попробовать всё это систематизировать, упростить и добавить новый функционал

Существующие решения

- Moodle, Schoology (многофункциональные, ориентированные на онлайн образование, как следствие, сложные, много что нужно настраивать, плохие приложения на android)
- LMS HSE (нет приложений, слишком официальный, медленное, постоянно разлогинивается)
- OpenLMS (есть приложение, медленное, плохой сервис поддержки, платное, постоянно разлогинивается)
- CoreAchieve (возможности интеграции, платное, сложная настройка)

Что хотим от приложения?

Большинство таких систем ориентировано на онлайн образование, а значит в них есть много функционала, который не применим или бесполезен для оффлайн. Этот функционал занимает UI, требует какой-то настройки и делает приложение слишком затратным по времени.

Мы хотим:

- Приложение-помощник для студентов и преподавателей
- Мало функционала, но он очень удобный
- Оптимизация под те сервисы и ту организацию процесса, которые используются на нашем направлении
- Возможность функционирования без администрации

Технологии

Сервер:

- Java + Kotlin
- Spring
- MySQL
- Docker
- Firebase Cloud Messaging
- Redis
- Google api services Sheets v4, Drive v3

Android приложение:

- Kotlin
- Firebase
- swagger-codegen 2 (OkHttp3, Retrofit2)
- Calendar Provider

Подзадачи. Самойлов Виктор

1. Спроектировать архитектуру БД и реализовать ее программно в виде набора POJO
   - Hibernate, ORM
2. Разработать классы для работы с базой данных
   - CrudRepository, Service, Transactions
3. Реализовать REST API для возможности взаимодействия внешнего мира с сервером
   - RestController, RequestMapping, Jackson, Swagger
4. Развернуть сервер удалённо и заменить in-memory database на постоянную
   - Yandex.Cloud, Docker-compose, MySQL
5. Реализовать отправку уведомлений с сервера
   - Firebase Cloud Messaging

Подзадачи. Венедиктов Роман

1. Спроектировать и написать основные Activities android клиента
   - xml, AppCompatActivity, ViewModel, Observer
2. Реализовать базовую аутентификацию пользователей
   - Firebase Authentication
3. Придумать способ отсылки уведомлений и реализовать их приём и обработку
   - Firebase Cloud Messaging
4. Реализовать REST клиент
   - swagger-codegen gradle plugin (Yelp/swagger-gradle-codegen)
5. Придумать формат данных для разбора таблиц, реализовать его, сделать так, чтобы это происходило при изменении данных
   - Drive Push Notifications + Sheets API

Подзадачи. Туров Кирилл

1. Написать скелет Activities android клиента
   - xml, AppCompatActivity, ViewModel
2. Изучить взаимодействие с календарем и реализовать добавления дедлайна
   - Google Calendar api, Android Calendar provider

Что получилось

Сервер:

- Реализована архитектура для базы данных, сервисы для работы с ней и REST API
- Есть сервер и база данных, которые развернуты удаленно
- Отправляются уведомления при изменении данных
- Работает простой парсинг таблиц и приём данных об их изменении

Android:

- Реализованы интерфейсы: логина, настроек, просмотра оценок, навигации
- Функционирует приём уведомлений
- Есть возможность создавать события в календаре

Планы на будущее

Сервер:

- Проанализировать производительность сервера, оптимизировать количество передаваемой по сети информации и количество SQL запросов к базе данных
- Улучшить парсинг таблиц, сформировать структуру данных для хранения структуры таблицы, выставить удобное REST API
- Добавить генерацию таблиц на основе сформированного формата парсинга

Android:

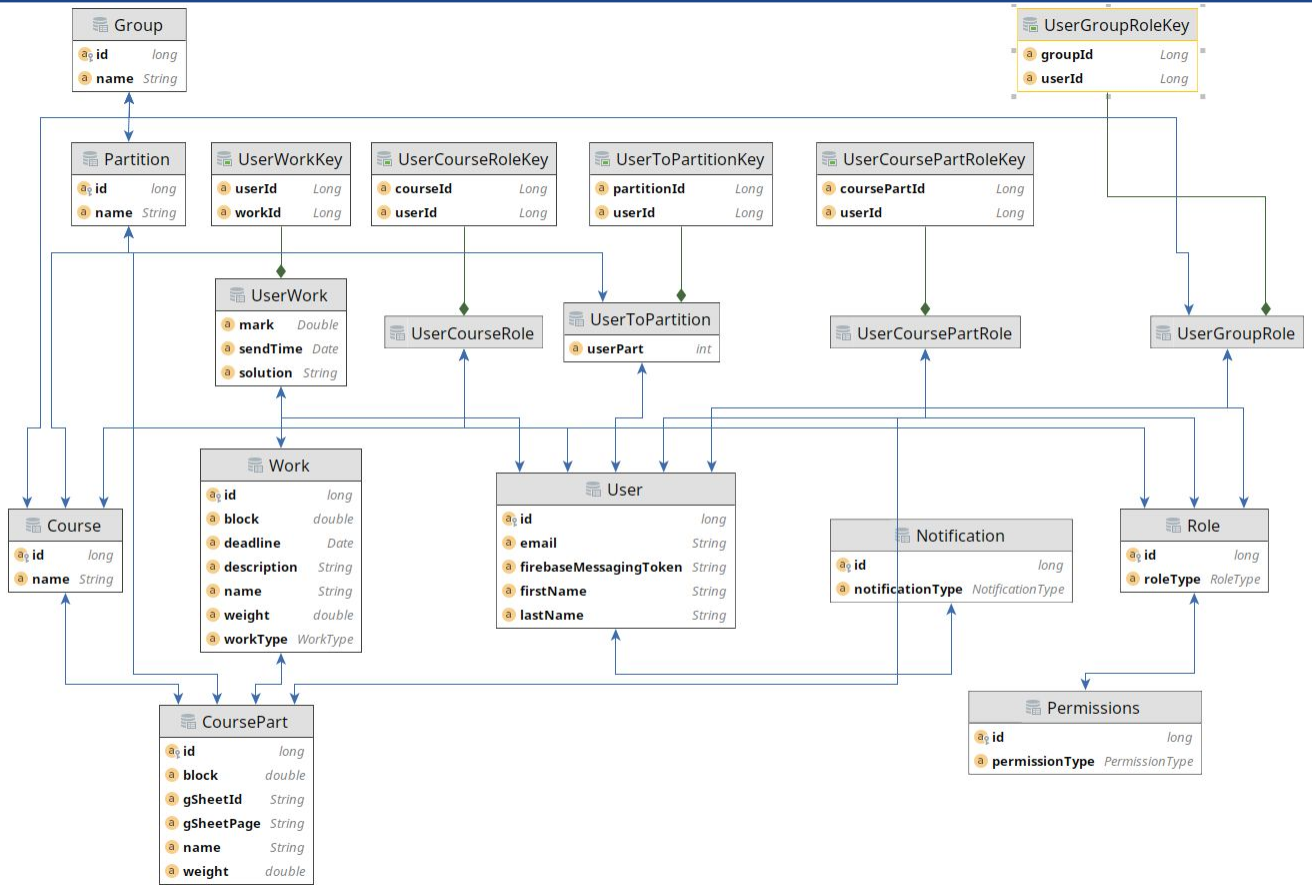
- Реализовать возможности администратора (добавление групп, курсов, таблиц...)

А также реализовать desktop и IOS клиентов

Ссылка на репозиторий

https://github.com/HSHelper/HSHelper

Архитектура БД



Итоговая архитектура