МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

СИСТЕМА КОНТРОЛЯ ПОСЕЩЕНИЙ ЗАНЯТИЙ.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ
ПО ДИСЦИПЛИНЕ «Операционные системы и системное программирование»

КП.ПО5.170154 - 04 81 00

Листов 17

Руководитель	Кочурко П.А
Выполнил	Корнасевич И. Д.
Консультант по ЕСПД	Кочурко П.А

Брест 2021

Содержание

Введение

С появлением компьютеров всё больше и больше документооборота стало переходить в электронный формат. Контроль посещаемости — не исключение.

Существует множество так называемых Content Management System (CRM), но в большинстве своём онн выполняют слишком общие задачи.

В данной работе будет разработана система контроля посещаемости занятий студентами с использованием клиент-серверной архитектуры, что позволит использовать эту систему не только на ПК, но и на мобильных устройствах.

1 Системный анализ и постановка задачи

Основной сценарий использования системы:

а) Все пользователи регистрируются в системе администратором.

б) Каждый пользователь имеет роль: преподаватель или студент.

в) Список занятий заносится в базу данных (БД) администратором.

г) Во время очередного занятия, преподавать на своём устройстве начинает занятие, у него появляется одноразовый QR-код, который сканируется студентами.

д) Студент, просканировавший QR-код считается посетившим занятие.

Из задачи, очевидно, что необходимо использовать какое-нибудь энергонезависимое хранилище, иначе получить статистику посещаемости просто невозможно.

Также необходимо написать WEB сервер, реализовать REST API на нём для взаимодействия с клиентом. Сервер также должен поддерживать WebSockets или Server Sent Events для обновления QR по инициативе сервера.

Клиент стоит сделать максимально простым и тонким. Для достижения кроссплатформенности максимально простым способом, проще всего сделать именно браузерный клиент. При разработке клиента стоит придерживаться Mobile First подхода, ввиду того, что большинство сценариев использования (сканирование QR кода, например), чаще всего производится посредством мобильного устройства.

К опциональным задачам стоит отнести:

- Вывод статистики по каждому конкретному студенту, преподавателю, предмету и т.п.
- Создание отдельного клиента для администратора

При реализации основного функционала, опциональные задачи не должны доставить особых проблем, ведь для их решения необходимо всего лишь расширить REST API.

2 Проектирование системы

Система в целом должна состоять из textbfклиента, textbfсервера и textbfБД. Общая схема отображена на рисунке 1.

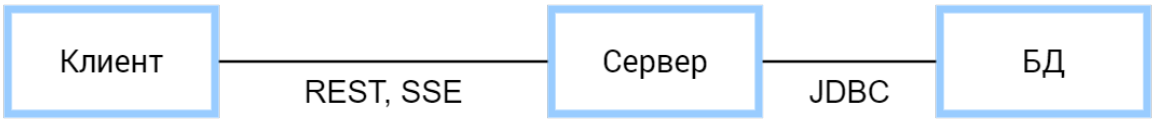

Рисунок 1 – Общая архитектура приложения.

2.1 Архитектура БД

Современные программные системы редко обходятся без энергонезависимого хранилища данных. Это могут быть реляционные, нереляционные (NoSQL), графовые БД. В конце концов данные можно хранить в простых текстовых файлах.

В разрабатываемой системе используется реляционная БД. Это самый простой, хорошо изученный и понятный способ хранить данные. В контексте этого приложения нет смысла использовать NoSQL решения, основные преимущества которых, а именно высокая гибкость, попросту бессмысленны в данной системе.

Как видно на рисунке 2, основные сущности: STUDENT, LESSON, TEACHER, STUDENT_GROUP.

- STUDENT_GROUP позволяет создавать группы студентов, при этом STUDENT и STUDENT_GROUP связаны отношением Many-to-many, что позволяет удобно назначать занятие сразу множеству студентов.
- STUDENT_APPOINTMENT хранит статус посещения занятия студентом.

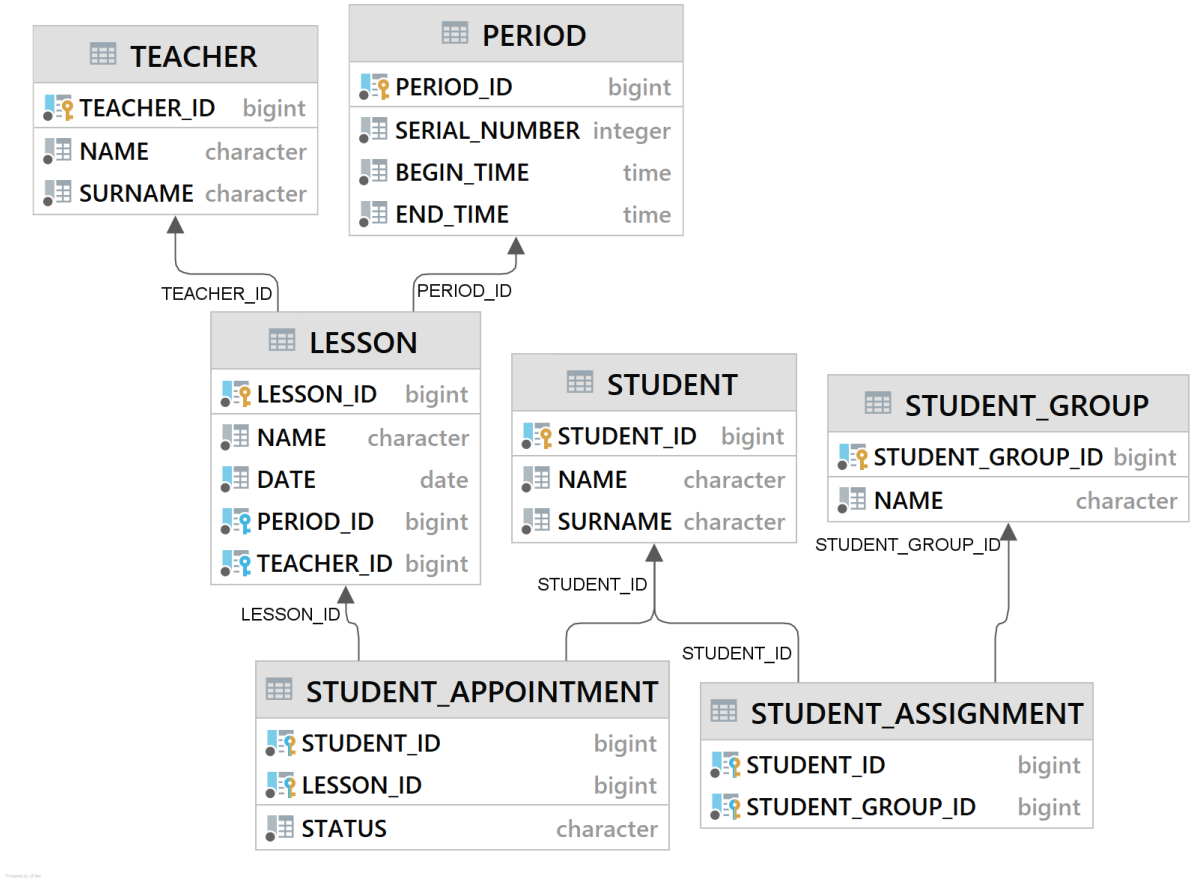

Рисунок 2 – Общая архитектура приложения.

– Каждый LESSON связан с PERIOD. PERIOD здесь выступает номером занятия (1-ая, 2-ая пара и т.д.)

В качестве конкретной БД использована H2, как простое in-memory решение, что сильно ускоряет разработку. В боевом применении лучше использовать H2 в Server или Mixed режиме, также хорошим выбором будет PostgreSQL.

2.2 Архитектура сервера

Серверная сторона должна отвечать следующим критериям:

– Достаточные познания меня, как разработчика, в выбранной платформе.

– Возможность легко взаимодействовать с реляционной БД

– Удобные инструменты реализации REST API

– Поддержка Server Sent Events (SSE)

– Доступные инструменты разработки

– Простота включения внешних библиотек в приложение

К критериям выше идеально подходит Java-Spring [4] платформа. Общая схема сервера на Spring выглядит следующим образом:

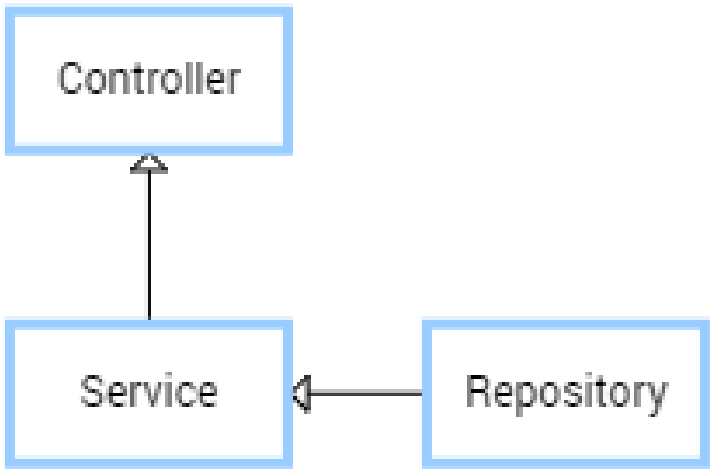

Рисунок 3 – Общая архитектура сервера.

– Controller формирует REST API, также отвечает за создание SSE. При получении HTTP запроса, в этом модуле формируются соответствующие Data Transfer Object (DTO), поля DTO валидируются, после управление передается в Service.

– В Service располагается логика приложения выполняет. Он обращается к Repository по необходимости. Также инфраструктурные задачи лежат в основном на Service.

– Repository отвечает на взаимодействие с БД. В этом слое объекты данные, превращаются в SQL запросы, а результаты этих запросов — в объекты Model.

2.2.1 Repository

Для взаимодействия с БД в Java обычно используют JPA, в частности Hibernate [1]. В Spring даже есть абстракция над JPA — Spring Data JPA [3]. Hibernate, как проверенный инструмент выбран в качестве инструмента работы с БД.

2.2.2 Build tools

Сборка современного Java приложение уже давно перестала быть тривиальной задачей. Причина тому — невероятное количество используемых библиотек. Для облегчения сборки специальные инструменты, такие как: Apache Ant, Apache Maven, Gradle

- Apache Ant был одним из первых инструментов автоматической сборки проектов для Java. Ant — императивный инструмент, поэтому каждое действие должно быть описано в файле конфигурации (в данном случае XML). Это придаёт невероятную гибкость во время сборки, но в тоже время, Ant конфигурация может быть невероятно длинной и сложной.
- Apache Maven в отличие от Ant — полностью декларативный инструмент. Начать новый проект на Maven очень просто, ведь он сам знает, как и куда загружать зависимости проекта, где находится XML конфигурация самого Maven и т.д. От программиста требуется всего-лишь следовать его ожиданиям. Слабая сторона Maven — кастомизация. Для задания дополнительной логики необходимо использовать Maven-плагины, которые не всегда полностью покрывают требования.
- Gradle находит баланс между императивностью Ant и декларативностью Maven. В Gradle отказались от XML заменив его Domain Specific Language (DSL). DSL позволяет декларативно описать сборку на Maven, а если нужно, то написать живой код на Groovy или Kotlin (каждый из которых является JVM языком).

Для проекта выбран Gradle, как одновременно простой, но гибкий инструмент. Gradle также предоставляет разбить цельное приложение на модули.

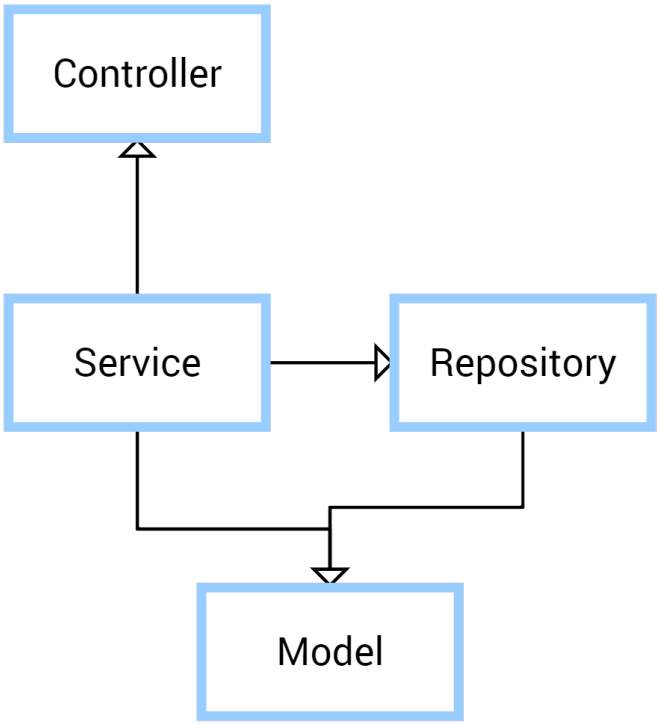

Рисунок 4 – Зависимости модулей друг от друга.

2.3 Архитектура клиента

На клиенте жизнь, в отличие от сервера, сильно проще.

Здесь уже есть стандартный пакетный менеджер npm.

Между Angular, View и React я выберу React, так как из вышеперечисленного я знаю только его.

В качестве языка программирования выбран JavaScript. Можно было выбрать TypeScript, но для небольшого проекта он породит больше проблем, чем решит.

Централизованное управление состоянием, такое как Redux или MobX, использовать вполне нужно. Для этой цели выбран Redux.

Для сканирования QR кода будет использована библиотека `react-qr-reader`, для его генерации — `qrcode.react`.

3 Реализация системы

3.1 Инструменты

Среда разработки — Intellij IDEA (Для сервера и клиента).
Сервер бэка — Embedded apache Tomcat.
Сервер фронта — Node.js.
Для написания пояснительной записки использован VS Code и Latex.

3.2 Сервер

На стороне сервера используется Gradle в качестве системы сборки. Однако, для запуска сервера отдельно устанавливать и настраивать окружение Gradle не нужно. Gradle-wrapper.jar идёт в комплекте с исходным кодом, что значительно упрощает развёртывание. Конфигурация Gradle в проекте представляет собой код на языке Kotlin.

3.2.1 Контроллеры

Каждый Rest контроллер — Spring bean, у которого методы помечены @RequestMapping (Может быть put, get и т.п.) Всего в проекте 3 контроллера:

а) CheckinController — отвечает за чекин студентов.

б) UserController — отвечает за авторизацию.

в) LessonMonitoringController — позволяет получить информацию о текущем занятии.

3.2.2 SSE

С проекте используются SSE, который прекрасно поддерживается в Spring. Всё, что нужно сделать, вернуть из контроллера объект типа SseEmitter. У этого объекта есть метод send, благодаря которому можно отправить данные на клиент.

Сервер работает с несколькими SSE одновременно. Поэтому, все SseEmitter-ы хранятся в оперативной памяти. В качестве коллекции выбран ConcurrentHashMap<Integer, CachingSseEmitter>, где ключ — id занятия, а значение — надстройка над обычными SseEmitter, запоминающая предыдущее сообщение, что позволяет использовать одноразовые QR коды.

3.2.3 Spring Data Jpa

Spring Data Jpa — часть стека Spring. Хотя это всего лишь надстройка над JPA, которая в свою очередь надстройка наж JDBC. По умолчанию Spring Data JPA использует Hibernate.

Сама концепция Hibernate подразумевает абстрагирование от реляционной БД при помощи построение двусвязного графа сущностей в Java. Эти сущности — Java объекты с конструктором по умолчанию, геттером и сеттером к каждому полю. Также они не могут быть final. Названия полей должны совпадать с названиями колонок (Hibernate сам умеет переводить camel-case в snake case). Сущности JPA обильно приправлены аннотациями (@OneToMany, @Entity и проч). По этим аннотациям Hibernate строит модель БД.

Для работы с БД в Spring Data JPA используется паттерн репозиторий. Согласно нему сущности не умеют сами себя сохранять в БД, но для этого существуют специальные объекты — репозитории. Они могут строить SQL на основе имени метода, HQL, и нативного SQL.

Вся работа с БД сосредоточена в модуле repo. Тут располагаются все JPA репозитории и сущности.

3.3 Клиент

React приложение создано при помощи create-react-app. Из него убраны все иконки React. Для оформления использован Matherial ui от Google.

Приложение делится на 3 модуля:

а) api — доступ к эндпоинтам (используется axios).

б) components — все React компоненты находятся здесь.

в) redux — здесь располагается управление состоянием приложения.

Клиент сам по себе делится на 2 части — для студента и преподавателя. Эти 2 части делят между собой форму логина.

В части преподавателя находится SSE части для генерации одноразового QR кода. В части студента — модуль сканирования QR кода и отправки хэша на сервер. При авторизации клиент получает информацию о роли пользователя, так он может выбрать вариант представления.

4 Тестирование

Spring предоставляет обширные возможности тестирования. Это и DataJPA тесты, и MockMVC тесты. Все тесты находятся в папках test или integration.

Для более полной картины, была создана тестовая БД (H2).

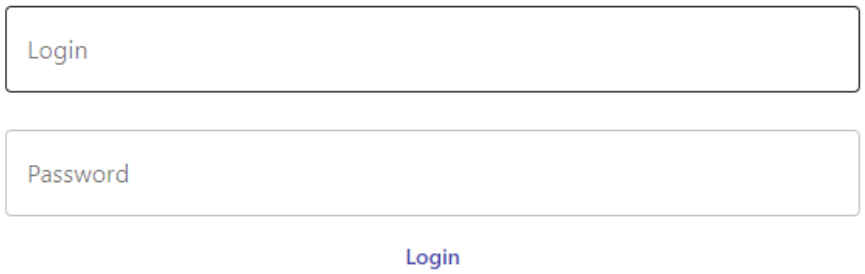

Рисунок 5 – Форма логина.

Рисунок 6 – Считывание QR кода.

Lesson begin 00:00:00
Lesson end 23:59:00
Students count 1

Рисунок 7 – Генерация QR-кода.

Lesson begin 00:00:00
Lesson end 23:59:00
Students count 1

Done

Lesson begin 00:00:00
Lesson end 23:59:00
Students count 1

Рисунок 8 – Код считан.

Заключение

Была разработана система контроля посещаемости занятий студентами. Система вполне выполняет поставленные требования.

В ходе выполнения работы были практически закреплены знания по работе с базами данных и их обработке, куда, к примеру, сохранялись данные о погоде набранного населённого пункта. В силу чего обеспечивалась быстрота работы приложения: база данных обслуживает сразу много потоков, поэтому приложением могут пользоваться одновременно сотни и тысячи человек.

Сервер реализован на Spring Framework (Java), в то время как клиентская часть на React—Redux.

Слабой частью получившейся системы является интерфейс, который получился достаточно скудным. Также не все опциональные задачи были выполнены (вроде создания админки).

В целом система справляется с основной задачей — контроль посещаемости.

Список литературы

1. Hibernate документация [электронный ресурс]. — URL: https://hibernate.org/orm/documentation/5.6/.
2. Spring Data JDBC документация [электронный ресурс]. — URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/#reference.
3. Spring Data JPA документация [электронный ресурс]. — URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference.
4. Spring документация [электронный ресурс]. — URL: https://docs.spring.io/spring-framework/docs/current/reference/html/.
5. ГОСТ 19.701-90 ЕСПД [электронный ресурс] : Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. — URL: https://znaytovar.ru/gost/2/GOST_1970190_ESPD_Sxemy_algori.html.
6. ГОСТ 7.1-2003 [электронный ресурс] : Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. — URL: https://www.internet-law.ru/gosts/gost/1560.