Geekbrains

**Разработка веб приложения «Рассписание Ярославского государственного технического университета» на JavaSpring**

Программа:

Разработчик

Специализация:

Веб-разработка на Java

Студент:

Яковлев А.В.

Ярославль

2024

**Содержание**

|  |  |
| --- | --- |
| Введение……………………………………………………………………. | 3 |
| Глава 1. Предпроектная стадия…………………………………………… | 5 |
| 1.1. Порядок разработки проекта………………………………………… | 5 |
| 1.2. Требования к приложению…………………………………………… | 6 |
| 1.3. Выбор технологий……………………………………………… | 7 |
| 1.4. Структура приложения………………………………………… | 8 |
| Глава 2. Проектная стадия………………………………………………… | 11 |
| 2.1. Проектирование общей архитектуры приложения………………… | 11 |
| Заключение………………………………………………………………… | 24 |
| Список использованной литературы…………………………………….. | 25 |

**Введение**

Большинство студентов ежедневно испытывают колоссальное напряжение от огромного количества каждодневных задач. Из-за подобного хаоса тяжело сосредоточиться на самом главном в студенческом деле – учебе. Решением данной проблемы является планирование своего распорядка дня и в частности учебы и расписания с домашними заданиями, курсовыми работами и их дедлайнами. В школе для этой цели использовали дневник, который должен быть у каждого школьника. Но в ВУЗе студенты предоставлены сами себе и сами должны контролировать свой распорядок.

Приложение «Расписание ЯГТУ» является удобным инструментом для отслеживания занятий в тот или иной день недели, домашних заданий и отзывах о преподавателях.

«Расписание ЯГТУ» предлагает удобный API для написания и изменения расписания, внесения и изменения домашних заданий и другие полезные «примочки» для студента.

Основная функциональность приложения:

- Расписание: Пользователь может просматривать и вносить изменения в существующее расписание.

- Домашние задания: Пользователь может просматривать и вносить изменения в домашние задания к предмету в определенный день недели.

- Преподаватели: Пользователь может просматривать информацию о преподавателе (его поведении, отношении к студентам и сложности сдачи ему зачета/экзамена) и вносить в нее изменения.

- Уведомления: Пользователь может получать уведомления о предстоящем учебном дне (какие будут предметы, какие д/з по ним в определенный день и в какой аудитории проводится занятии)

- Принадлежность: Каждый пользователь при регистрации сможет выбрать факультет, курс и группу в которой он обучается и сможет просматривать кто обучается вместе с ним.

Цель данной работы: упростить насыщенную студенческую жизнь и помочь в освоении учебного материала.

Данное приложение было разработано студентом направления Веб-разработка на Java Яковлевым А.М.

1. **Предпроектная стадия**

**1.1 Порядок разработки проекта**

Проект был разработан в следующем порядке:

**- Планирование проекта**

* Определение основных функций используемых в приложении
* Выбор используемых технологий
* Построение архитектуры приложения

**- Установка необходимого ПО**

* Установка IDE Intellij Idea с необходимыми плагинами
* Скачивание JavaDevelopmentKit
* Установка ApacheMaven (сборщик проектов)
* Установка SpringBoot

- **Создание проекта**

* Выбор необходимых Spring зависимостей

- **Разработка**

* создание модели данных
* создание репозитория
* создание иерархии приложения
* создание сервиса
* написание документации

- **Дебаг и оптимизация**

* Отладка кода для корректной работы приложения
* Оптимизация кода для улучшения читабельности и ускорения работы приложения

**1.2 Требования к приложению**

Для разработки приложения «Расписание ЯГТУ» были определены следующие требования:

- Функциональные требования:

* Пользователь должен иметь возможность просматривать, редактировать и создавать расписание
* Пользователь должен иметь возможность просматривать, редактировать и создавать факультеты, группы и преподавателей
* Пользователь должен иметь возможность просматривать, редактировать и создавать домашние задания
* Пользователь должен иметь возможность устанавливать и изменять состояние предмета в расписании (запланировано/отменено)

- Нефункциональные требования:

* Интерфейс приложения должен быть максимально интуитивным для пользователя
* Интерфейс должен быть максимально информативным и содержать достаточное количество информации для понимания
* Данные пользователей должны храниться и передаваться в зашифрованном виде
* Приложение должно иметь хорошую производительность
* Приложение должно поддерживать возможность увеличения числа пользователей без ухудшения производительности
* Обновления для поддержания функциональности и безопасности приложения

**1.3 Выбор технологий**

**- Язык программирования:** Для разработки серверного приложения отлично подходит язык программирования Java. Он позволяет создавать масштабируемые приложения. За счет своей кроссплатформенности Java позволяет в будущем модернизировать приложения путем создания его мобильных версий, виджетов и плагинов.

- **Framework:** Spring Boot – упрощает настройку и развертывание приложения, предоставляет встроенные серверы и автоматическую конфигурацию. Spring Data JPA – упрощает работу с базами данных. Spring Security – обеспечивает безопасность приложения, включая аутентификацию и авторизацию пользователей.

- **Бэкенд технологии:** H2 – реляционная база данных, используемая для хранения данных о пользователях, расписаниях, курсах и преподавателях.

Hibernate – ORM (объектно-реляционное отображение) – инструмент, который используется для работы с базой данных через объекты java.

- **Фронтенд технологии:** Thymeleaf – шаблонизатор java, который позволяет создавать динамические веб-страницы. HTML/CSS/JavaScript – основные технологии для создание пользовательского интерфейса.

- **Аутентификация и авторизация:**

Spring Security применяется для обеспечения безопасности веб-приложения, а также для аутентификации пользователей и доступа к ресурсам.

- **Среда разработки:** Для разработки серверной части приложения используется самая популярная на сегодняшний день IDE для разработки java приложений – Intellij Idea.

- **Инструменты сборки и управления зависимостями:** Maven или Gradle. В нашем приложении используется ApacheMaven

- **Тестирование:** Для тестирования приложения используется фреймворк Junit, предназначенный для автоматического тестирования программ.

- **Контейнеризация:** Docker используется для контейнеризации приложения и упрощение его развертывания на компьютере.

- **Система контроля версий:** Для контроля версий используется GIT.

- **Сервер приложений:** Для развертывания веб-приложения используется Apache Tomcat.

**1.4 Структура приложения**

Архитектура приложения реализуется паттерном MVC(model-view-controller) и включает в себя следующие основные компоненты:

- **Модель (Model)**

Модель отвечает за управление данными приложения и бизнес-логикой. Она взаимодействует с базой данных и определяет, как данные будут храниться, обрабатываться и передаваться.

- Классы сущностей: определяют структуру данных: Student, Course, Teacher, Schedule

- Уровень доступа к данным: использует ORM (например, Hibernate) для взаимодействия с базой данных. Сюда могут входить репозитории или DAO (объекты доступа к данным).

- Бизнес-логика: методы, которые обрабатывают данные, например, создание, обновление, удаление расписаний, проверка данных и т. д.

- **Представление (View)**

Представление отвечает за отображение данных пользователю и взаимодействие с ним. Оно получает данные от контроллера и отображает их в удобном для пользователя формате.

- HTML/CSS: используется для разметки и стилизации веб-страниц.

- JavaScript: для добавления интерактивности на стороне клиента (например, динамическое обновление расписания без перезагрузки страницы).

- Шаблоны: если используется серверный рендеринг, могут быть использованы шаблоны (Например Thymeleaf).

**- Контроллер (Controller)**

Контроллер служит посредником между моделью и представлением. Он обрабатывает входящие запросы от пользователя, взаимодействует с моделью для получения или изменения данных и передаёт результаты в представление.

- Обработка запросов: контроллеры принимают HTTP-запросы (например, GET, POST, PUT, DELETE) и определяют, какие действия необходимо выполнить.

- Взаимодействие с моделью: контроллер вызывает методы модели для получения или изменения данных.

- Передача данных в представление: контроллер передаёт данные в представление для отображения пользователю.

- **Репозиторий (Repository):**

Компонент «Репозиторий» (Repository) в архитектуре программного обеспечения выполняет несколько ключевых функций, связанных с управлением данными. Он служит абстракцией для взаимодействия с источником данных (например, базой данных) и обеспечивает удобный интерфейс для выполнения операций с данными.

**- Сервис (Service):**

В архитектуре MVC (Model-View-Controller) компонент «Сервис» (или сервисный уровень) выполняет несколько ключевых функций:

- Бизнес-логика: обработка данных, выполнение вычислений и применение правил, которые определяют, как должны обрабатываться данные.

- Инкапсуляция: сервисы инкапсулируют логику, связанную с работой с данными. Это позволяет отделить бизнес-логику от представления (View) и контроллеров (Controller), что упрощает тестирование и поддержку кода.

- Работа с данными: сервисы могут взаимодействовать с репозиториями (или DAO — объектами доступа к данным), которые отвечают за доступ к данным. Это позволяет абстрагироваться от конкретных источников данных (например, баз данных, API и т. д.).

- **Сервер (Server):**

Это контроллер, который обеспечивает связь между пользователем и приложением, представлением и моделью. Он получает запросы от пользователя, обрабатывает их и передаёт данные модели для обработки.

- Обеспечивает связь между пользователем и приложением, представлением и моделью.

- Получает запросы от пользователя, обрабатывает их и передаёт данные модели для обработки.

- Выбирает представление для отображения пользователю и передает ему необходимые данные.

**2 Проектная стадия**

**2.1 Проектирование общей архитектуры приложения**

Для начала разработки приложения "расписание университета" на Java с использованием Spring, Maven и базы данных, лучше всего начать с проектирования структуры проекта и создания основных компонентов.

Сначала создадим maven проект:

Используя IDE Intellij Idea создаем новый проект со сборщиком maven. При создании указываем GroupId «ru.yakovlev» и ArtefactId «university-schedule».

Затем нужно настроить файл pom.xml. Добавить в него зависимости Spring boot, Thymeleaf и другие необходимые библиотеки.

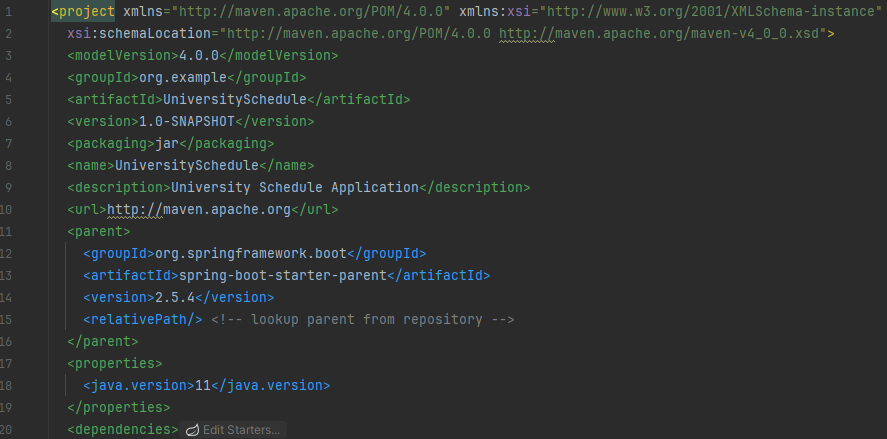


Рисунок 1 – Код файла pom.xml

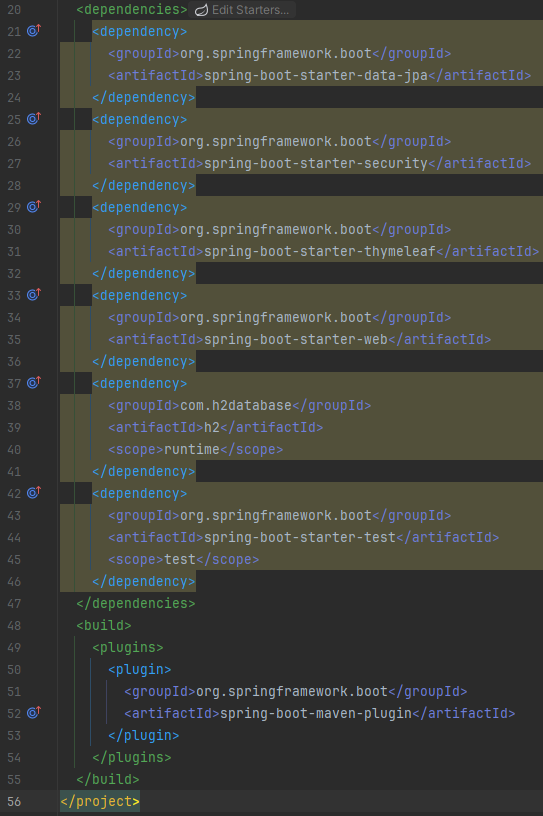


Рисунок 2 – Окончание кода файла pom.xml

Следующим шагом будет настройка Spring Boot.

Создадим главный класс приложения UniversitySheduleApplication.java в директории src\main\java\ru\yakovlev\universityschedule.

UniversitySheduleApplication.java является главным классом, содержащим метод main, который запускает Spring Boot приложение.

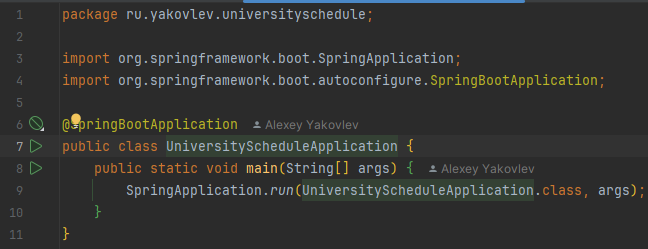


Рисунок 3 – Код класса UniversitySheduleApplication.java

Следующим шагом является создание моделей. Здесь создаются модели для пользователей, расписания, преподавателей и предметов.

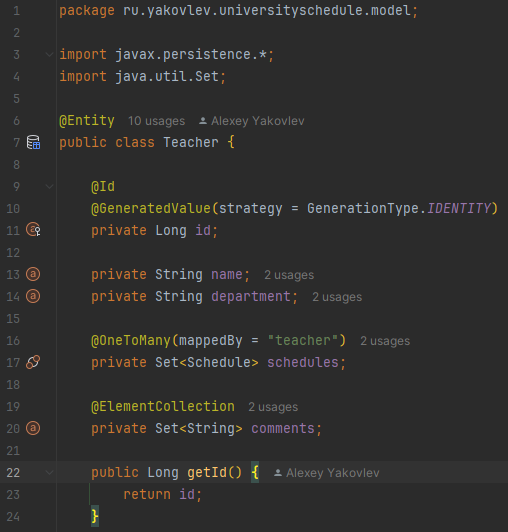


Рисунок 4 – Часть кода модели Teacher

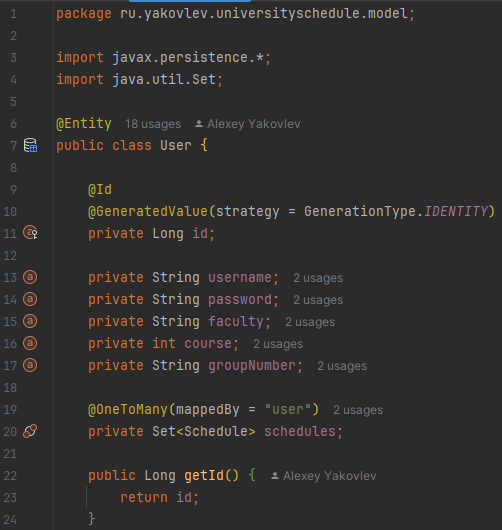


Рисунок 5 – Часть кода модели User

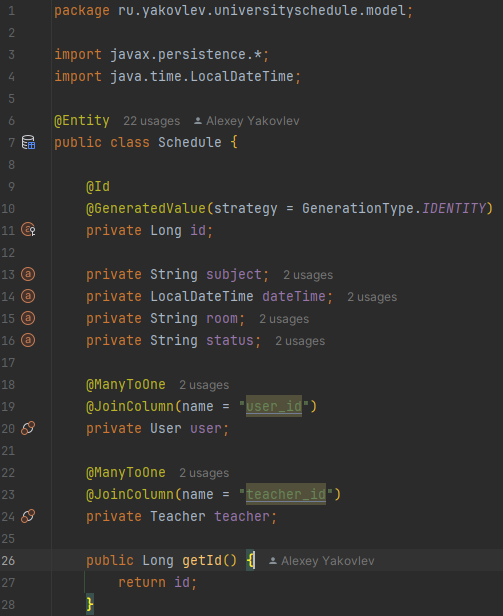


Рисунок 6 – Часть кода модели Schedule

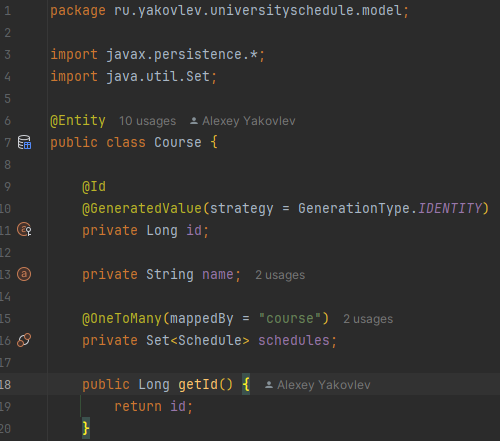


Рисунок 7 – Часть кода модели Course

Каждая модель будет представлять собой таблицу в базе данных.

Следующим шагом будет создание репозиториев для взаимодействия с базой данных. Репозитории представлены интерфейсами и для их создания используется Spring Data JPA.

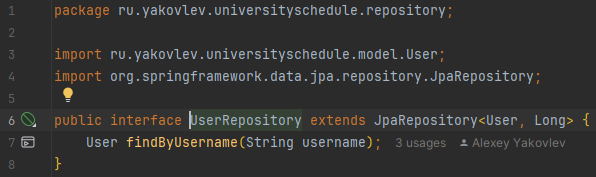


Рисунок 8 – Код репозитория UserRepository

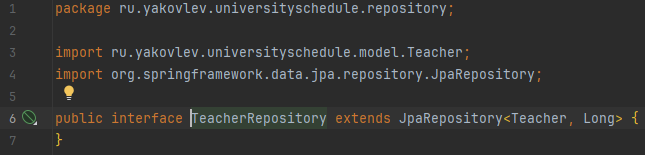


Рисунок 9 – Код репозитория TeacherRepository

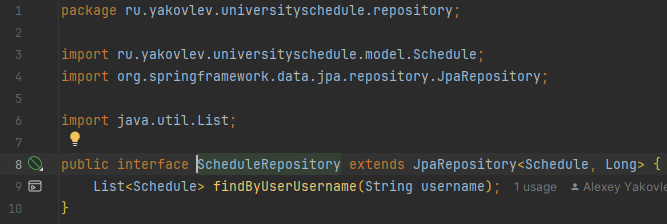


Рисунок 10 – Код репозитория ScheduleRepository

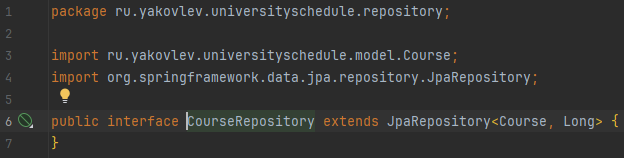


Рисунок 11 - Код репозитория CourseRepository

Следующим шагом в разработке приложения будет создание сервисов. Сервисы будут содержать основную бизнес-логику приложения.

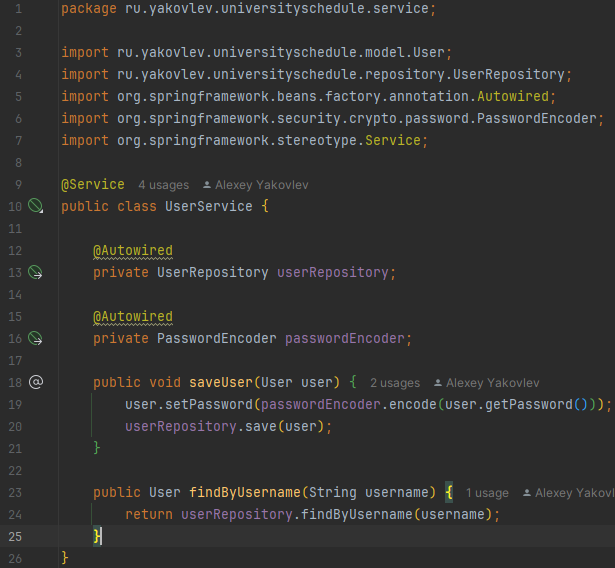


Рисунок 12 – Код сервиса UserService

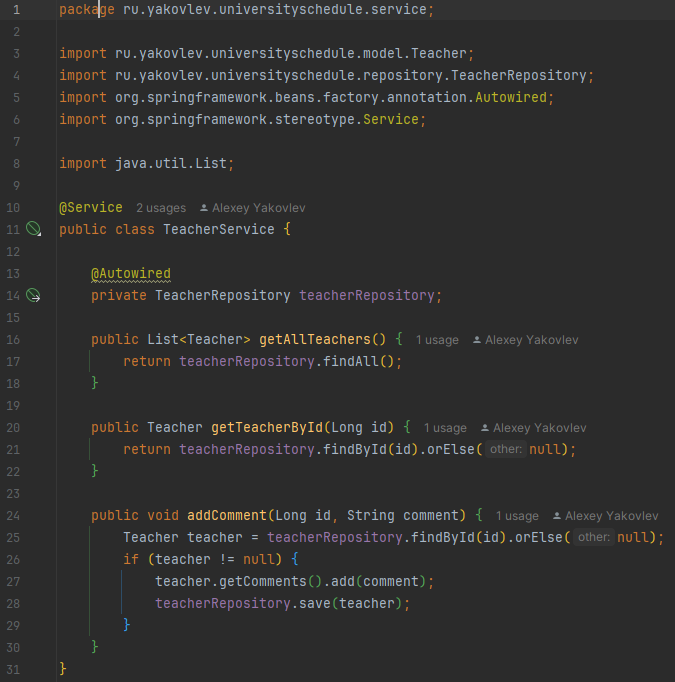


Рисунок 13 – Код сервиса TeacherService

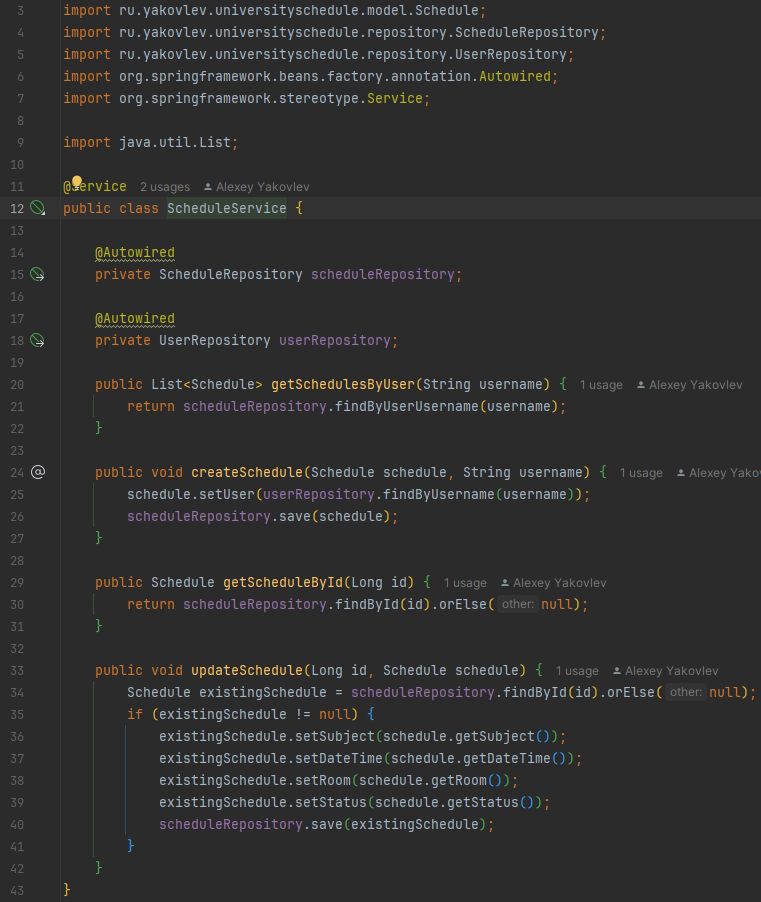


Рисунок 14 – Код сервиса ScheduleService

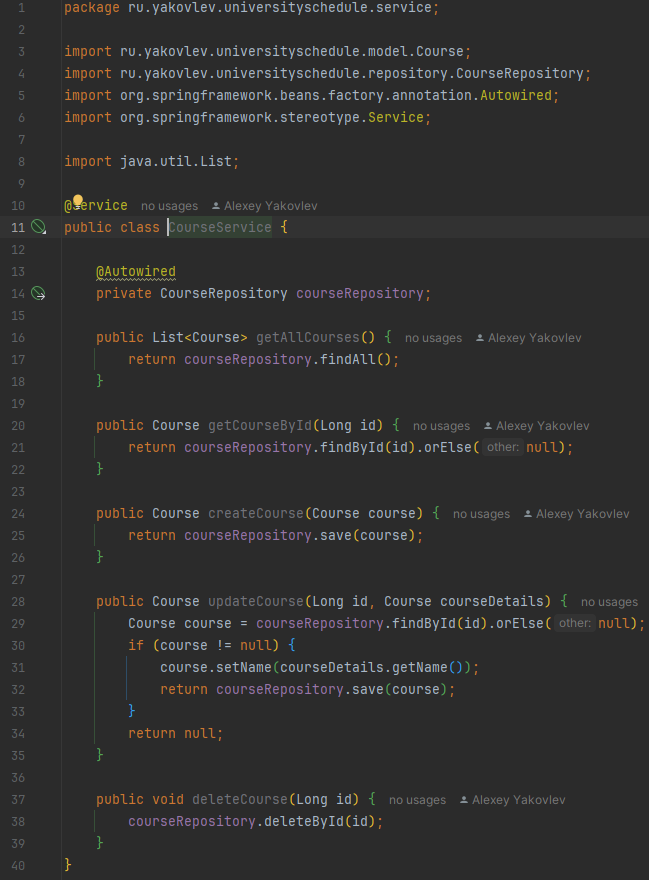


Рисунок 15 – Код сервиса CourseRepository

Следующим шагом является написание контроллеров. Контроллеры будут обрабатывать запросы от пользователей и возвращать ответы.

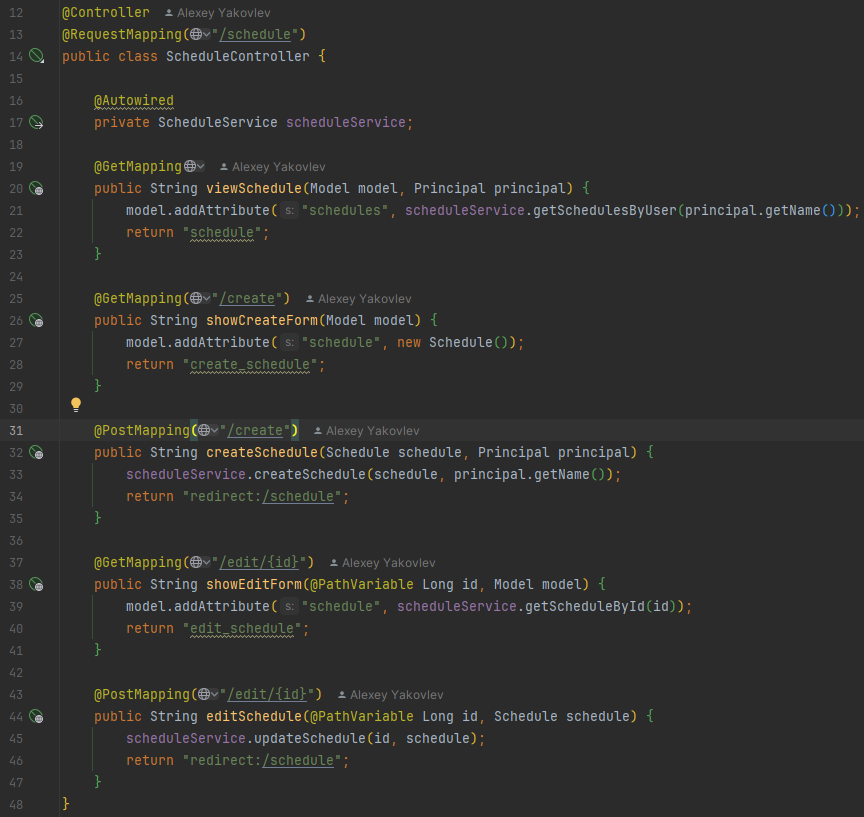


Рисунок 16 – Код контроллера ScheduleController

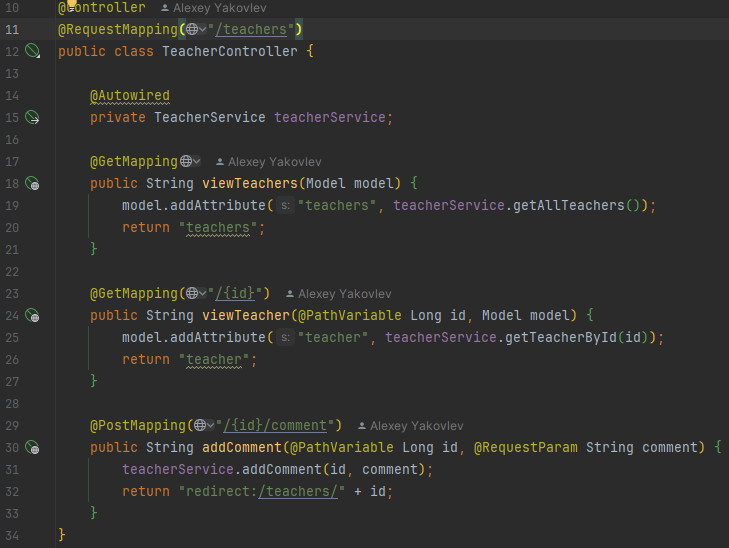


Рисунок 17 – Код контроллера TeacherController

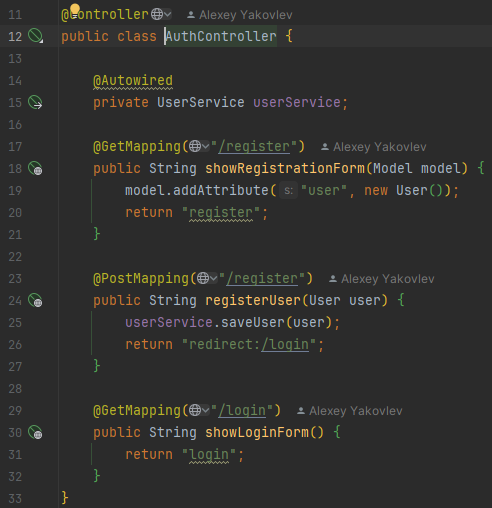


Рисунок 18 – Код контроллера AuthController

Далее следует настройка безопасности приложения. Для этого нужно настроить Spring Security для аутентификации и авторизации для защиты приложения.

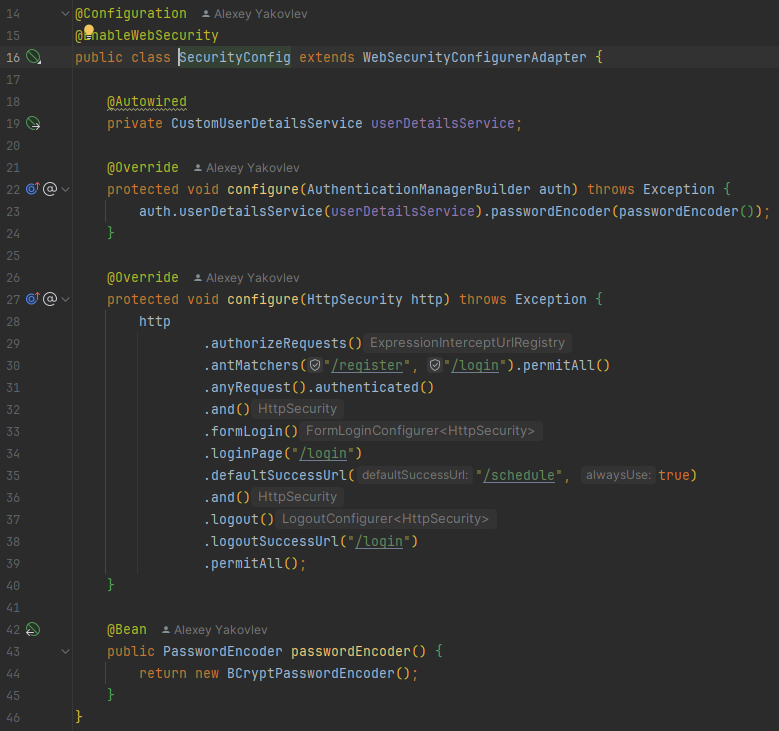


Рисунок 19 – Код класса SecurityConfig

Далее идет описание базы данных

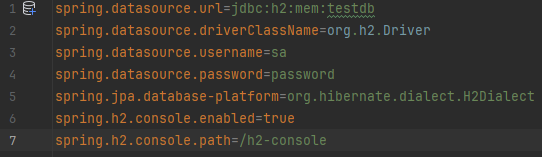


Рисунок 20 – Код файла application.properties

Для тестирования приложения пишем файл UniversityScheduleApplicationTests.java в котором используем библиотеку Jupiter.

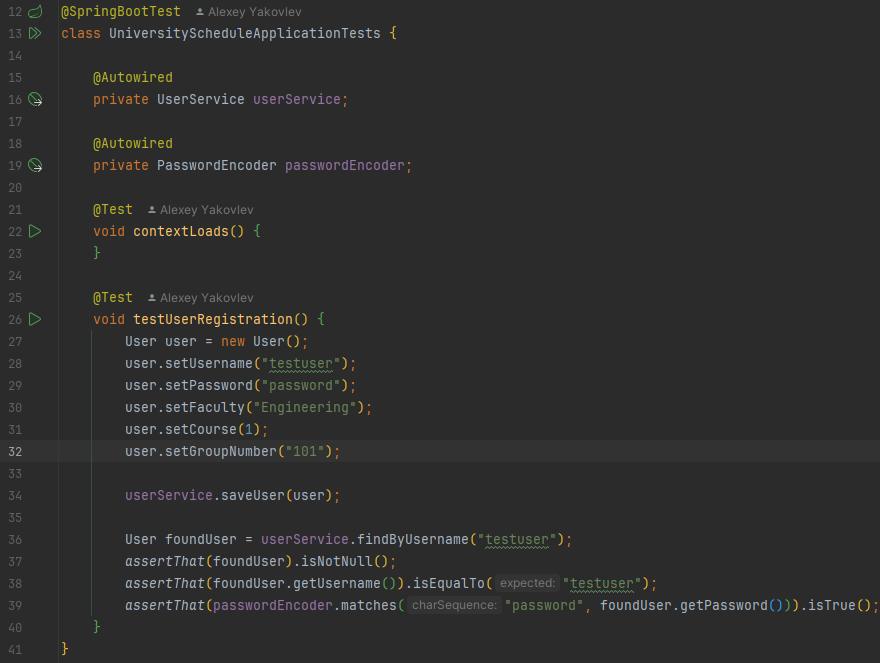


Рисунок 21 – Код файла UniversityScheduleApplicationTests

Завершающим штрихом является запуск приложения с помощью команды «mvn spring-boot:run»

**Заключение**

В рамках выпускной квалификационной работы было разработано приложение «Расписание университета ЯГТУ», предназначенное для упрощения студенческой жизни Ярославского политеха.

Проект был осуществлен с использование современных IT технологий таких как: Java, Spring, Maven, H2 и других технологий и библиотек.

Дальнейшее развитие приложения может идти в направлении улучшения интуитивности интерфейса, добавление нового функционала (например: выбор световой темы) и перемещение приложения на мобильные платформы.

**Список литературы**

1. Учебный материал GeekBrains (<https://gb.ru/education>)
2. Учебные видеоматериалы по Spring от Alishev
3. Статьи и видеоматериалы из сети интернет
4. Учебный материал JavaRush