МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ “ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ” (ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Веб-Платформа для удаленного обучения музыке

Курсовая работа

09.03.04 Программная инженерия

Профиль « Информационные системы и сетевые технологии »

Зав. кафедрой _____ *С.Д. Махортов, д.ф.- м.н., доцент*__.__.20__

Обучающийся _____*М.О.З. Тавфик, 3 курс*

Руководитель _____*А.А. Вахтин, к.ф-м.н., доцент*

Воронеж 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире технологические инновации переплетаются с различными сферами человеческой деятельности, привнося в них новые возможности и перспективы развития. Одной из таких сфер является образование, которое неизменно подвергается влиянию цифровой революции. Вместе с тем, развитие образовательных технологий позволяет создавать новые платформы и сервисы, способствующие более эффективному и доступному обучению. В этом контексте рассматривается создание платформы для удаленного обучения музыке, охватывающей как учителей, так и студентов, с целью обеспечения эффективного обучения и обмена знаниями в музыкальной сфере.

Образование стало одной из важнейших сфер деятельности человечества, оказывающей существенное влияние на формирование и развитие личности, а также на социальное и экономическое развитие общества в целом. С развитием информационных технологий образовательные процессы приобретают новые формы и методы, становясь более гибкими, доступными и персонализированными. Особенно важно это в контексте обучения музыке, где необходимо обеспечить не только теоретические знания, но и практические навыки, взаимодействие с учителями и коллегами, а также доступ к разнообразным образовательным ресурсам.

Целью данной работы является разработка и создание платформы для удаленного обучения музыке, которая позволит учителям создавать курсы, а студентам — проходить обучение и отправлять задания. Для достижения этой цели поставлены следующие задачи:

- Исследование существующих подходов к удаленному обучению музыке и анализ современных образовательных технологий.

- Разработка концепции и функциональных требований к платформе для удаленного обучения музыке.
- Проектирование архитектуры и выбор технологических стеков для реализации платформы.
- Создание и тестирование прототипа платформы.

Оценка эффективности и потенциала платформы для обучения музыке.

Платформа для удаленного обучения музыке представляет собой инновационный подход к организации образовательного процесса в музыкальной сфере. Разработка такой платформы требует не только интеграции современных информационных технологий, но и учета специфики обучения музыке, включая возможности для онлайн-взаимодействия, обратной связи и практических занятий.

Создание платформы для удаленного обучения музыке имеет большую практическую значимость для образовательного сообщества, музыкальных школ, учреждений и студентов. Платформа предоставляет возможность учителям музыки расширить свою аудиторию и проводить курсы удаленно, а студентам — получить доступ к качественному обучению, независимо от места жительства или времени.

В работе представлен анализ существующих подходов к удаленному обучению музыке, разработка концепции и функциональных требований к платформе, описание архитектуры и реализация прототипа платформы, а также оценка ее эффективности и потенциала для обучения музыке. Каждая часть работы раскрывает определенные аспекты разработки и реализации платформы, обеспечивая комплексный подход к решению поставленных задач.

1. Постановка задачи

1.1. Цели создания приложения

Целями создания веб-приложения MuseSpark являются:

- Расширение доступа к музыкальному образованию.
- Удобство для преподавателей и студентов.
- Улучшение качества образовательного процесса.

1.2. Требования к разрабатываемой системе

1.2.1. Функциональные требования

- Регистрация и аутентификация пользователей.
- Управление курсами.
- Управление заданиями.
- Прохождение курсов и выполнение заданий.
- Роли пользователей и авторизация.
- Уведомления и оповещения.
- Безопасность данных.

1.2.2. Технические требования

- Технологический стек.
- Развертывание и инфраструктура.
- Безопасность.
- Производительность и масштабируемость.
- API и интеграции.
- Логирование и мониторинг.

1.3. Требования к интерфейсу

- Интуитивно понятный дизайн - Интерфейс должен быть простым и легким для навигации как для студентов, так и для преподавателей. Все основные функции и действия должны быть легко доступны и понятны пользователям без необходимости в подробной инструкции.
- Адаптивный интерфейс - Платформа должна корректно отображаться на различных устройствах и экранах, включая настольные компьютеры, ноутбуки, планшеты и смартфоны. Все элементы интерфейса должны адаптироваться под размер

экрана, чтобы обеспечить удобство использования на любом устройстве.

- Разделение интерфейса для разных типов пользователей - Интерфейс должен учитывать различия в потребностях преподавателей и студентов. Преподаватели должны иметь доступ к инструментам для создания и управления курсами, а студенты – к инструментам для прохождения курсов и сдачи заданий. Это должно быть реализовано через отдельные панели или страницы для каждой роли.
- Интерактивные элементы и визуальная обратная связь - Все интерактивные элементы, такие как кнопки, формы и ссылки, должны предоставлять визуальную обратную связь при взаимодействии. Это может включать изменение цвета, появление анимаций или всплывающих подсказок, чтобы пользователи могли уверенно взаимодействовать с интерфейсом.
- Система уведомлений - Интерфейс должен включать систему уведомлений для информирования пользователей о важных событиях, таких как новые задания, изменения в расписании курсов, результаты оценок и т.д. Уведомления должны быть видимы и легко доступны, чтобы пользователи всегда были в курсе актуальной информации.

1.4. Задачи, решаемые в процессе разработки

- Проектирование базы данных - Разработка структуры базы данных для хранения информации о пользователях, курсах, уроках, заданиях и других сущностях, необходимых для функционирования платформы. Это включает определение связей между таблицами, выбор подходящих типов данных и оптимизацию производительности запросов.
- Разработка функционала регистрации и аутентификации - Создание механизмов для регистрации новых пользователей, хранения и проверки учетных данных, а также для аутентификации зарегистрированных пользователей при входе

на платформу. Это также включает обработку сбоев в аутентификации и предотвращение утечки информации.

- Разработка управления курсами и заданиями - Реализация функционала для преподавателей, позволяющего создавать, редактировать и управлять курсами, включая добавление новых уроков, создание заданий и установку сроков сдачи. Также необходимо обеспечить возможность просмотра и оценки выполненных заданий студентами.
- Разработка интерфейса для прохождения курсов - Создание интерфейса для студентов, позволяющего легко просматривать доступные курсы, проходить уроки, сдавать задания и отслеживать свой прогресс. Это также включает поддержку механизмов обратной связи, например, возможность задавать вопросы преподавателям и просматривать обсуждения курсовых тем.
- Обеспечение безопасности приложения - Разработка механизмов обработки и защиты от возможных угроз безопасности, таких как атаки переполнения буфера, инъекции SQL, межсайтового скриптинга (XSS) и других уязвимостей. Это также включает реализацию мер аутентификации, авторизации и контроля доступа для защиты конфиденциальной информации пользователей.

2. Анализ предметной области

2.1. Терминология (глоссарий) предметной области

- Платформа для удаленного обучения: Веб-приложение, предназначенное для обучения пользователей различным предметам и навыкам через онлайн-курсы и задания.
- Курс: Структурированный набор учебных материалов, представленных в виде последовательности уроков или модулей, направленных на обучение определенному предмету или навыку.
- Урок: Отдельный блок учебного материала внутри курса, представленный в виде видео, текста, аудио или других

форматов и содержащий информацию и инструкции для студентов.

- Задание: Учебная задача или проект, предложенный для выполнения студентами в рамках курса с целью практического применения знаний и навыков, полученных на уроках.
- Преподаватель: Пользователь платформы, который создает и управляет курсами, добавляет учебный материал, назначает задания и взаимодействует со студентами.
- Студент: Пользователь платформы, который зарегистрировался на курсе для прохождения обучения, просмотра уроков, выполнения заданий и получения обратной связи от преподавателей.
- Регистрация: Процесс создания учетной записи на платформе для удаленного обучения путем предоставления персональной информации и выбора учетных данных.
- Сессия: Временный период, в течение которого пользователь взаимодействует с платформой для удаленного обучения, обычно начинается при входе пользователя на сайт и заканчивается при его выходе.
- Интерфейс пользователя (UI): Часть приложения, предназначенная для взаимодействия пользователя с системой, включая элементы управления, графические элементы и макеты.
- База данных: Структурированное хранилище данных, используемое для сохранения информации о пользователях, курсах, уроках, заданиях и других сущностях, связанных с платформой для удаленного обучения.

2.2. Обзор аналогов

На текущем рынке существует несколько аналогичных платформ для удаленного обучения с различными функциями и особенностями. Ниже приведен краткий обзор некоторых из них:

- Coursera: Крупная платформа для обучения онлайн, предлагающая широкий выбор курсов от университетов и организаций по всему миру. Coursera предоставляет доступ к лекциям, упражнениям, тестам и сертификатам.
- Udemy: Это онлайн-рынок для обучения и обучающих курсов, созданных как профессионалами, так и любителями. Udemy предлагает курсы по различным темам, включая технические навыки, бизнес, личное развитие и многое другое.
- Skillshare: Платформа для обучения онлайн, специализирующаяся на курсах по творчеству, дизайну, бизнесу и техническим навыкам. Skillshare позволяет пользователям создавать и делиться своими собственными курсами.
- LinkedIn Learning: Это платформа для обучения онлайн, интегрированная с социальной сетью LinkedIn. LinkedIn Learning предлагает более 16 000 курсов по бизнесу, техническим навыкам, мягким навыкам и многому другому.

2.3. Моделирование системы

2.3.1. Диаграммы вариантов использования

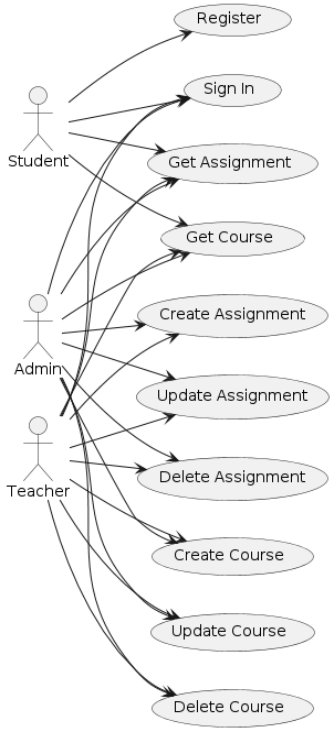

Рисунок 1 - Диаграммы вариантов использования

Диаграмма вариантов использования (Use Case Diagram) представляет собой графическое описание функциональности системы с точки зрения ее пользователей. В данном случае представлены основные варианты использования для различных типов пользователей: студентов, учителей и администраторов.

2.3.2. Диаграмма классов

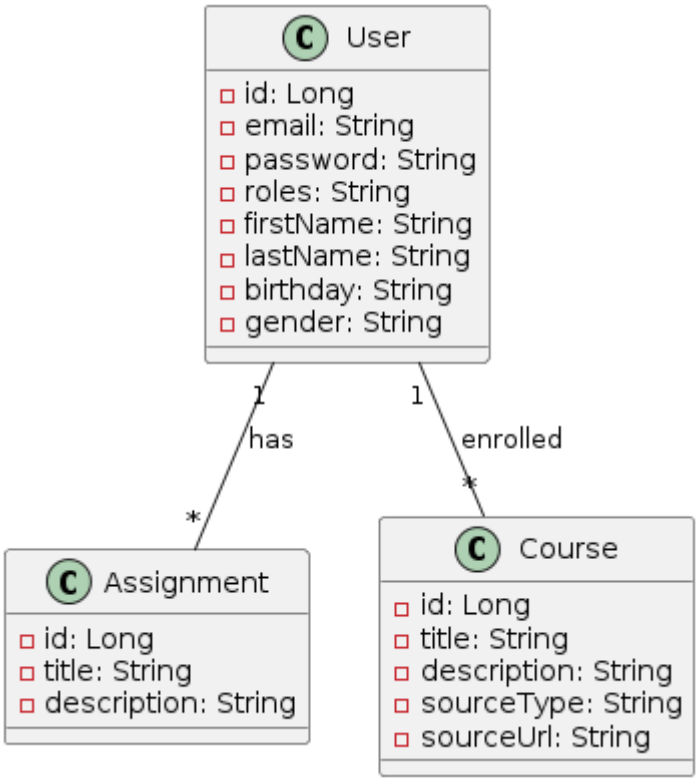

Рисунок 2 - Диаграмма классов

Диаграмма классов представляет собой структурное описание системы, отображающее классы, их атрибуты и связи между ними.

На данной диаграмме представлены основные классы системы:

- User (Пользователь): Класс, описывающий пользователей системы. У пользователя есть ряд атрибутов, таких как идентификатор (id), электронная почта (email), пароль (password), роли (roles), имя (firstName), фамилия (lastName), дата рождения

(birthday) и пол (gender). Пользователи имеют отношение "has" с заданиями (Assignment) и "enrolled" с курсами (Course), указывая на то, что пользователи могут иметь несколько заданий и курсов.

- Assignment (Задание): Класс, описывающий задания в системе. Задание имеет атрибуты id (идентификатор), title (название) и description (описание).
- Course (Курс): Класс, описывающий курсы в системе. Курс имеет атрибуты id (идентификатор), title (название), description (описание), sourceType (тип источника курса) и sourceUrl (URL-адрес источника курса).

Связи между классами отражают основные отношения между объектами. Например, каждый пользователь может иметь несколько заданий и курсов, что отображается связями "1" -- "*" между классом User и классами Assignment и Course соответственно.

2.3.3. Диаграмма последовательностей

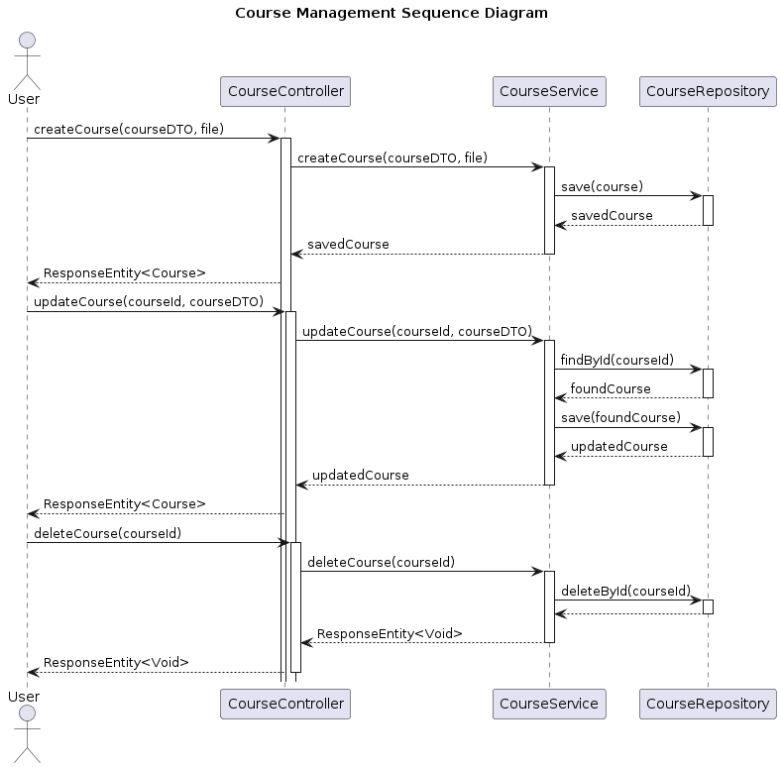

Рисунок 3 - Диаграмма последовательностей

Диаграммы последовательностей описывают взаимодействие между объектами в рамках определенной последовательности действий или вызовов методов. В данном случае представлены диаграммы последовательностей для операций создания, обновления и удаления курсов в системе управления курсами.

2.3.4. Диаграмма состояний настройки

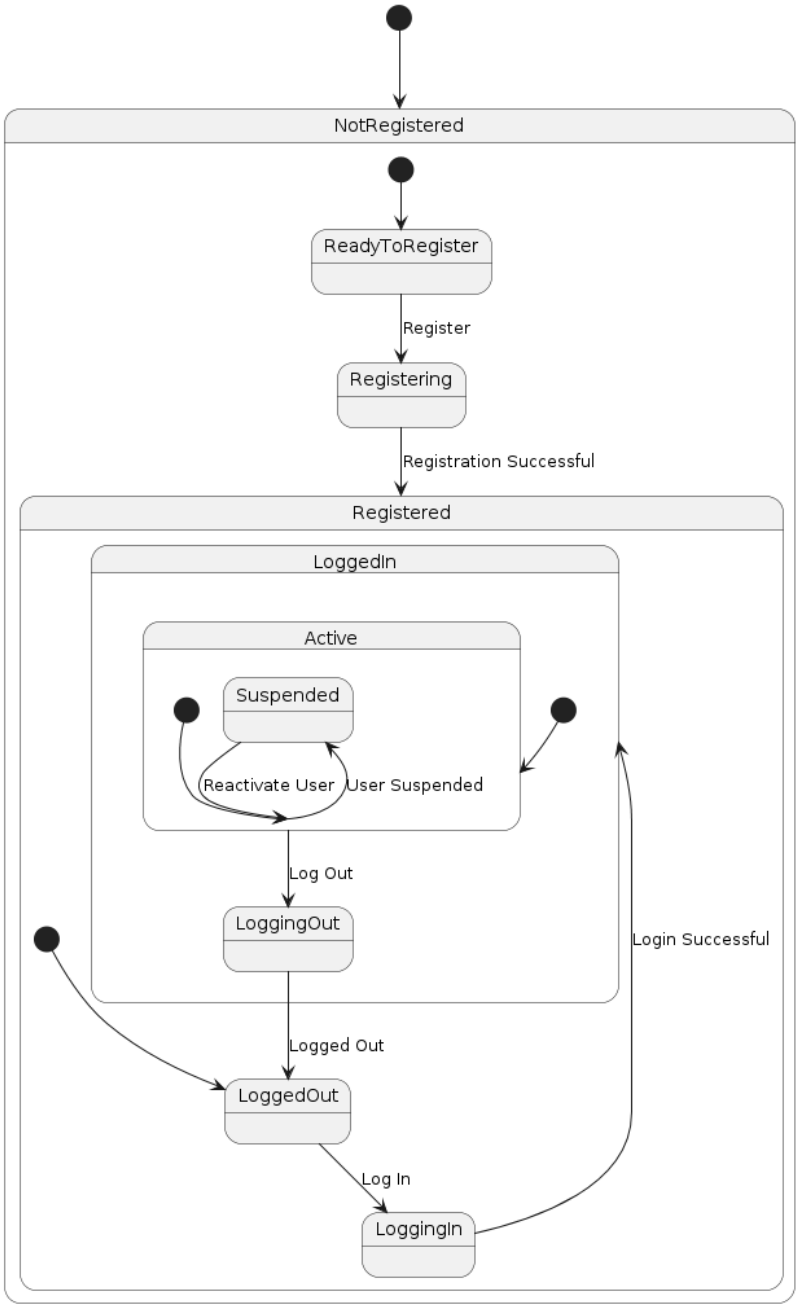

Рисунок 4 - Диаграмма состояний настройки

Диаграмма состояний настройки показывает различные состояния, через которые проходит пользователь при взаимодействии с системой в контексте регистрации, входа в систему и активности учетной записи.

3. Реализация

3.1. Средства реализации

Для реализации проекта MuseSpark используются различные технологии как на стороне клиента (frontend), так и на стороне сервера (backend). Ниже представлены основные инструменты и технологии, применяемые в проекте:

Backend:

- Java: Язык программирования, используемый для разработки серверной части приложения.
- Spring Boot: Фреймворк для создания веб-приложений на языке Java. Он предоставляет мощные средства для создания RESTful API и управления зависимостями.
- MySQL: Реляционная база данных, используемая для хранения данных приложения. MySQL широко распространен и поддерживается множеством инструментов.

Frontend:

- React: JavaScript-библиотека для создания пользовательских интерфейсов. React облегчает разработку интерактивных и масштабируемых пользовательских интерфейсов.
- JavaScript (JS): Язык программирования, который используется для добавления динамического поведения на веб-страницах.
- HTML: Язык разметки, используемый для создания структуры веб-страниц.
- CSS: Язык таблиц стилей, который используется для оформления и стилизации веб-страниц.

Объяснение выбора:

- Java и Spring Boot: Java предоставляет высокую производительность, надежность и масштабируемость. Spring

Boot упрощает разработку и развертывание веб-приложений Java.

- MySQL: MySQL - это надежная и широко используемая реляционная база данных, которая отлично подходит для хранения структурированных данных приложения.
- React, JavaScript, HTML, CSS: React обеспечивает быстрое рендеринг и управление компонентами пользовательского интерфейса. JavaScript, HTML и CSS являются основными технологиями для разработки интерактивных веб-интерфейсов.

3.2. Обзор альтернативных средств реализации

Помимо используемых средств реализации, существуют и альтернативные технологии, которые также могут быть использованы для создания приложения MuseSpark:

Backend:

- Языки программирования: Вместо Java можно рассмотреть другие языки, такие как Python (с фреймворком Django или Flask), Node.js (с фреймворком Express) или Ruby (с фреймворком Ruby on Rails).
- Фреймворки: Вместо Spring Boot можно использовать фреймворки для других языков, например, Flask или Django для Python, Express для Node.js и Ruby on Rails для Ruby.
- Системы управления базами данных (СУБД): Вместо MySQL можно выбрать другие реляционные СУБД, такие как PostgreSQL или SQLite, или даже нереляционные базы данных, такие как MongoDB.

Frontend:

- Фреймворки и библиотеки: Помимо React, можно рассмотреть другие фреймворки и библиотеки JavaScript, такие как Angular или Vue.js.
- Препроцессоры CSS: Для более эффективного написания стилей CSS можно использовать препроцессоры, такие как Sass или Less.

- Языки разметки: Помимо HTML, можно использовать более продвинутые языки разметки, такие как JSX (используемый в React) или Pug.

Разработка мобильных приложений:

- Для создания мобильных приложений можно использовать фреймворки и инструменты, такие как React Native (для разработки кросс-платформенных мобильных приложений), Flutter (для создания кросс-платформенных мобильных приложений) или нативную разработку для Android (с использованием Java или Kotlin) и iOS (с использованием Swift или Objective-C).

3.3. Реализация интерфейса

Реализация интерфейса веб-приложения MuseSpark может включать в себя следующие шаги:

Frontend:

- Разработка макета и дизайна: Создание дизайна пользовательского интерфейса, включая макеты страниц, элементы управления, цветовую схему и т. д.
- Верстка HTML и CSS: На основе дизайна создание структуры HTML-разметки и стилей CSS для каждой страницы и компонента интерфейса.
- Интеграция с бэкендом: Настройка взаимодействия с бэкендом с помощью API, написание функционала для отправки и получения данных.
- Реализация клиентской логики: Написание JavaScript-кода для реализации функциональности взаимодействия пользователя с интерфейсом, включая валидацию форм, анимации и другие пользовательские действия.
- Тестирование интерфейса: Проведение тестирования пользовательского интерфейса для выявления и устранения ошибок, проверка работы на разных устройствах и браузерах.

Backend:

- Настройка серверной части: Создание и настройка сервера с использованием Spring Boot, настройка маршрутов и контроллеров для обработки запросов от клиентской части.
- Работа с базой данных: Настройка соединения с базой данных MySQL, создание моделей данных (Entity классы) и репозиториев для взаимодействия с базой данных.
- Бизнес-логика: Написание сервисов и компонентов, реализующих бизнес-логику приложения, такую как создание и обновление курсов и заданий, аутентификация и авторизация пользователей и т. д.
- Аутентификация и безопасность: Настройка механизмов аутентификации и авторизации пользователей с использованием Spring Security, реализация защиты ресурсов и API.
- Тестирование бэкенда: Проведение модульного и интеграционного тестирования бэкенд-кода для проверки его корректности и надежности.

Интеграция и тестирование:

- Интеграция frontend и backend: Обеспечение взаимодействия между клиентской и серверной частями приложения, тестирование API и передачи данных.
- Общее тестирование: Проведение тестирования всего приложения в целом для проверки его работоспособности, корректности работы интерфейса и функциональности.

4. Тестирование

4.1. Тестирование серверной части

Тестирование серверной части приложения MuseSpark, реализованной на Java с использованием Spring Boot и MySQL, может включать в себя следующие этапы:

- Модульное тестирование сервисов: Написание и запуск модульных тестов для каждого сервиса, который содержит бизнес-логику приложения. В этих тестах можно проверить

различные сценарии использования методов сервисов, их корректность и обработку исключений.

- Интеграционное тестирование контроллеров: Написание тестов для проверки взаимодействия контроллеров с сервисами. Это включает отправку HTTP-запросов к API приложения и проверку соответствующих HTTP-ответов, а также проверку корректности передачи данных между контроллерами и сервисами.
- Тестирование базы данных: Написание и запуск тестов, проверяющих корректность взаимодействия с базой данных. В этих тестах можно проверить создание, чтение, обновление и удаление данных в базе данных, а также проверить корректность работы запросов SQL.
- Тестирование аутентификации и авторизации: Написание тестов для проверки механизмов аутентификации и авторизации, реализованных с использованием Spring Security. В этих тестах можно проверить корректность работы механизмов аутентификации пользователей, а также проверить доступ к защищенным ресурсам приложения.
- Общее интеграционное тестирование: Проведение тестирования всей серверной части приложения в целом для проверки ее работоспособности в реальных условиях. В этих тестах можно проверить взаимодействие всех компонентов приложения, а также проверить его корректность и надежность при работе с реальными данными.

4.2. Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса (UI) приложения MuseSpark, реализованного с использованием React, JavaScript, HTML и CSS, включает в себя следующие этапы:

- Модульное тестирование компонентов React: Написание модульных тестов для каждого компонента пользовательского интерфейса. Эти тесты могут проверять отображение компонента, его взаимодействие с пользователем и обработку событий.

- Интеграционное тестирование страниц: Написание тестов для проверки взаимодействия между компонентами на страницах приложения. Это включает тестирование навигации между страницами, передачу данных между компонентами и обработку их изменений.
- Тестирование пользовательского ввода: Проведение тестирования пользовательского ввода, такого как заполнение форм, отправка данных и обработка ошибок. В этих тестах можно проверить корректность работы валидации данных, обработку ошибок и корректность взаимодействия с API серверной части приложения.
- Тестирование респонсивности и кросс-браузерности: Проверка отображения пользовательского интерфейса на различных устройствах и в различных браузерах для обеспечения его корректной работы и отображения. Это включает тестирование на мобильных устройствах, планшетах, настольных компьютерах и в различных версиях браузеров.
- Общее интеграционное тестирование UI: Проведение тестирования всего пользовательского интерфейса в целом для проверки его работы в реальных условиях. В этих тестах можно проверить взаимодействие всех компонентов UI, а также проверить его корректность и надежность при работе с реальными данными и действиями пользователя.

После проведения тестирования пользовательского интерфейса приложения MuseSpark можно быть уверенным в его надежности, корректности работы и удобстве использования для конечных пользователей.

ЗАКЛЮЧЕНИЕ

Проект MuseSpark представляет собой важную платформу для обучения музыке, объединяющую учителей и студентов, а также облегчающую процесс создания курсов и выполнения заданий. Разработка данного проекта позволила создать мощный инструмент, способствующий распространению знаний и улучшению образования в сфере музыки.

Основные результаты работы над проектом включают в себя:

- Создание полнофункциональной серверной части на базе Java, Spring Boot и MySQL, обеспечивающей управление курсами, заданиями и пользователями.
- Реализация дружественного и интуитивно понятного пользовательского интерфейса на основе React, JavaScript, HTML и CSS, обеспечивающего удобство использования и навигации для всех типов пользователей.
- Разработка системы аутентификации и авторизации с использованием JWT (JSON Web Tokens) для обеспечения безопасности данных и защиты конфиденциальности пользователей.
- Проведение тестирования как серверной, так и пользовательской части приложения для обеспечения его надежной работы и соответствия требованиям.

Проект MuseSpark имеет большой потенциал для дальнейшего развития и расширения функциональности. Внедрение дополнительных возможностей, таких как расширенные инструменты аналитики, интерактивные учебные материалы и механизмы обратной связи, может значительно повысить его ценность и привлекательность для пользователей.

В целом, проект MuseSpark представляет собой важный вклад в область образования и музыкальной индустрии, и его успешное развитие может принести значительную пользу как учителям, так и студентам в их образовательном пути.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. React.js Official Documentation [Электронный ресурс]. – Режим доступа: https://reactjs.org/docs/getting-started.html - Заглавие с экрана. (Дата обращения: 08.06.2024).
2. Node.js официальная документация [Электронный ресурс]. – Режим доступа: https://nodejs.org/en/docs/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
3. Express.js Framework Documentation [Электронный ресурс]. – Режим доступа: https://expressjs.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
4. MySQL Official Documentation [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
5. HTML5 Rocks - A resource for open web HTML5 developers [Электронный ресурс]. Режим доступа: https://www.html5rocks.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
6. CSS Tricks - Tips, Tricks, and Techniques on using Cascading Style Sheets [Электронный ресурс]. – Режим доступа: https://css-tricks.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
7. SpringBoot Official Documentation [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot Заглавие с экрана. (Дата обращения: 08.06.2024).

ПРИЛОЖЕНИЕ

```
package com.martinosama.musespark.Controller;

import com.martinosama.musespark.DTO.AssignmentDTO;
import com.martinosama.musespark.Entity.Assignment;
import com.martinosama.musespark.Service.AssignmentService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/assignments")
public class AssignmentController {
    private final AssignmentService assignmentService;

    public AssignmentController(AssignmentService assignmentService) {
        this.assignmentService = assignmentService;
    }

    @PostMapping
    public ResponseEntity<Assignment> createAssignment(@RequestBody
AssignmentDTO assignmentDTO) {
        Assignment assignment =
assignmentService.createAssignment(assignmentDTO);
        return new ResponseEntity<>(assignment, HttpStatus.CREATED);
    }

    @GetMapping("/{assignmentId}")
    public ResponseEntity<Assignment> getAssignmentById(@PathVariable
Long assignmentId) {
        Assignment assignment =
assignmentService.getAssignmentById(assignmentId);
        if (assignment != null) {
            return ResponseEntity.ok(assignment);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PutMapping("/{assignmentId}")
    public ResponseEntity<Assignment> updateAssignment(@PathVariable
Long assignmentId, @RequestBody AssignmentDTO assignmentDTO) {
        Assignment updatedAssignment =
assignmentService.updateAssignment(assignmentId, assignmentDTO);
        if (updatedAssignment != null) {
            return ResponseEntity.ok(updatedAssignment);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{assignmentId}")
    public ResponseEntity<Void> deleteAssignment(@PathVariable Long
assignmentId) {
        assignmentService.deleteAssignment(assignmentId);
```

```
        return ResponseEntity.noContent().build();
    }
}
```

```
package com.martinosama.musespark.Controller;

import com.martinosama.musespark.DTO.CourseDTO;
import com.martinosama.musespark.Entity.Course;
import com.martinosama.musespark.Service.CourseService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

@RestController
@RequestMapping("/courses")
public class CourseController {
    private final CourseService courseService;

    public CourseController(CourseService courseService) {
        this.courseService = courseService;
    }

    @PostMapping(consumes = {"multipart/form-data"})
    public ResponseEntity<Course>
createCourse(@RequestPart("courseDTO") CourseDTO courseDTO,
                                                @RequestPart(value =
"file", required = false) MultipartFile file) {
        try {
            Course course = courseService.createCourse(courseDTO,
file);
            return new ResponseEntity<>(course, HttpStatus.CREATED);
        } catch (IOException e) {
            e.printStackTrace();
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
        }
    }

    @GetMapping("/{courseId}")
    public ResponseEntity<Course> getCourseById(@PathVariable Long
courseId) {
        Course course = courseService.getCourseById(courseId);
        if (course != null) {
            return ResponseEntity.ok(course);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PutMapping("/{courseId}")
    public ResponseEntity<Course> updateCourse(@PathVariable Long
courseId, @RequestBody CourseDTO courseDTO) {
        Course updatedCourse = courseService.updateCourse(courseId,
courseDTO);
        if (updatedCourse != null) {
```

```
            return ResponseEntity.ok(updatedCourse);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{courseId}")
    public ResponseEntity<Void> deleteCourse(@PathVariable Long
courseId) {
        courseService.deleteCourse(courseId);
        return ResponseEntity.noContent().build();
    }
}
```

```
package com.martinosama.musespark.Controller;

import com.martinosama.musespark.DTO.UserDTO;
import com.martinosama.musespark.Entity.User;
import com.martinosama.musespark.Service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.core.userdetails.UsernameNotFoundException
;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;
import java.util.logging.Logger;

@Controller
@RequestMapping("/api")
public class UserController {

    private static final Logger logger =
Logger.getLogger(UserController.class.getName());

    @Autowired
    private UserService userService;

    @PostMapping("/register")
    @ResponseBody
    public ResponseEntity<?> registerUser(@RequestBody UserDTO userDTO)
{
        logger.info("Attempting to register user: " +
userDTO.getEmail());
        Optional<User> existingUser =
userService.findByEmail(userDTO.getEmail());
        if (existingUser.isPresent()) {
            logger.warning("User already exists: " +
userDTO.getEmail());
            return ResponseEntity.badRequest().body("User already
exists");
        }
        try {
            User user = userService.createUser(userDTO);
            logger.info("User registered successfully: " +
```

```
user.getEmail());
            return new ResponseEntity<>(user, HttpStatus.CREATED);
        } catch (Exception e) {
            logger.severe("Error occurred while registering user: " +
e.getMessage());
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Registrat
ion failed");
        }
    }

    @PostMapping("/signin")
    @ResponseBody
    public ResponseEntity<?> signInUser(@RequestBody UserDTO userDTO) {
        try {
            User user = userService.signInUser(userDTO.getEmail(),
userDTO.getPassword());
            return ResponseEntity.ok(user);
        } catch (UsernameNotFoundException ex) {
            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid username
or password");
        }
    }
}
```