МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

|  |  |
| --- | --- |
| Институт | информатики и кибернетики |
| Кафедра | геоинформатики и информационной безопасности |

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Технология построения защищенных распределенных приложений»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Сайт для онлайн-тестирования по математике для детей»

|  |  |
| --- | --- |
| Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | М.Э. Волков |
| Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | А.Н. Дубина |
| Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | А.В. Сергеев |
|  |  |

САМАРА 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

|  |  |
| --- | --- |
| Институт | информатики и кибернетики |
| Кафедра | геоинформатики и информационной безопасности |

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Студентам ***Дубина Андрею Николаевичу*** гр.6511 и ***Волкову Максиму Эрнестовичу*** гр.6512

Тема проекта: ***«*Сервисное приложение для курьеров*»***

|  |  |  |
| --- | --- | --- |
| Планируемые результаты освоения образовательной программы (компетенции) | Планируемые результаты практики | Содержание задания |
| ОПК-3 способность применять языки, системы и инструментальные средства программирования в профессиональной деятельности  ПСК-7.1 способность разрабатывать и исследовать модели информационно-технологических ресурсов, разрабатывать модели угроз и модели нарушителя информационной безопасности в распределенных информационных системах | Знать: основные виды программных средств, технологию разработки алгоритмов и программ и методы их отладки, основы системного программирования и объектно-ориентированного подхода к программированию;  Уметь: использовать средства ОС для решения собственных задач, обосновывать проектные решения по структуре базы данных, работать с современными системами программирования, самостоятельно осваивать новые программные средства;  Владеть: навыками работы с различными операционными системами, методиками составления SQL- запросов и разработкой инфологической и логической моделей предметной области, языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ.  знать: современную классификацию средств защиты информации в корпоративных вычислительных сетях и системах, инструментальные программно-аппаратные средства анализа защищенности информационных систем и сетей, технологии моделирования угроз безопасности информации при проектировании распределенных автоматизированных систем в защищенном исполнении;  уметь: разрабатывать модели информационно-технологических ресурсов, модели угроз и модели нарушителя информационной безопасности в распределенных информационных системах, применять современные средства защиты информации при аудите распределенных компьютерных систем, применять современные технологии моделирования угроз безопасности информации при проектировании распределенных автоматизированных систем в защищенном исполнении;  владеть: навыками разработки моделей угроз и моделей нарушителей информационной безопасности процессов создания и эксплуатации распределенных автоматизированных систем в защищенном исполнении; определения угроз безопасности информации, потенциально и/или реально существующие в процессе создания и эксплуатации автоматизированных систем; применения современных технологий моделирования угроз безопасности информации при проектировании распределенных автоматизированных систем в защищенном исполнении; анализа достаточности мер по определению угроз безопасности информации в процессе создания и эксплуатации распределенных автоматизированных систем в защищенном исполнении. | Создать защищенный, удобный и функциональный сайт для проведения математических тестов, ориентированный на школьников младших классов.  Основные задачи:   1. Разработка структуры и функционала сайта для онлайн тестирования. 2. Создание удобного интерфейса для пользователей (учеников и администраторов). 3. Обеспечение безопасности данных пользователей и системы в целом. 4. Отладка и тестирование функционала для устойчивой работы в условиях реального использования. |

Дата выдачи задания 12 сентября 2024 г.

Срок представления на кафедру отчета о практике 24 декабря 2024 г.

Руководитель курсового проекта

доцент каф. ГИиИБ, к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сергеев А.В.

*(подпись)*

Задание принял к исполнению

студент группы № 6511-100503D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дубина А.Н.

*(подпись)*

Задание принял к исполнению

студент группы № 6512-100503D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волков М.Э.

*(подпись)*

РЕФЕРАТ

**Пояснительная записка к курсовому проекту:** 43 с,8 рисунков, 5 источников.

ВЕБ-ПРИЛОЖЕНИЕ, ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, БЕЗОПАСНОСТЬ ДАННЫХ, РОЛИ ПОЛЬЗОВАТЕЛЕЙ.

Цель работы – предоставить детям удобную платформу для изучения математики и проверки своих знаний с помощью сайтом для онлайн-тестирования по математике для детей.

Платформа для онлайн-тестирования по математике должна предоставлять возможность регистрации и авторизации пользователей, включая детей, родителей и учителей.

Каждый пользователь будет иметь доступ к определенному набору функций в зависимости от своей роли. Родители смогут просматривать прогресс ребенка, настраивать сложность заданий и получать отчеты.

Учителя получат инструменты для создания и мониторинга выполнения заданий, а также возможность отслеживать результаты группы учащихся. Дети смогут выполнять задания, ориентированные на устный счет и таблицу умножения, с возможностью самопроверки и мгновенной обратной связи.

Интерфейс платформы должен быть простым, интуитивно понятным и адаптированным для детей младшего школьного возраста.

Поддержка различных устройств, включая компьютеры, планшеты и смартфоны, обеспечит доступность системы в любое время.

Платформа должна быть гибкой, масштабируемой и готовой к дальнейшему развитию для удовлетворения образовательных потребностей.

СОДЕРЖАНИЕ

[ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ 2](#_Toc185184766)

[ВВЕДЕНИЕ 8](#_Toc185184766)

[1.ВЫБРАННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ](#_Toc185184767) 10

[2.ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ](#_Toc185184768) 12

[**2.1.Структура программы**](#_Toc185184769) 12

[**2.2. Описание основного кода** 15](#_Toc185184770)

[**2.2.1.** **Папка** **«ru.test\_service»** 15](#_Toc185184771)

[**2.2.1.1.** **Модуль «authorization»** 15](#_Toc185184772)

[**2.2.1.1.1**. **Модуль «configuration»** 15](#_Toc185184773)

[**2.2.1.1.2**. **Модуль «controller»:** 16](#_Toc185184774)

[**2.2.1.1.3**. **Модуль «dto»:** 17](#_Toc185184775)

[**2.2.1.1.4**. **Модуль «exception»:** 18](#_Toc185184776)

[**2.2.1.1.5**. **Модуль «service»:**](#_Toc185184777) 19

[**2.2.1.2.** **Модуль «common»**](#_Toc185184778) 19

[**2.2.1.2.1**. **Модуль «configuration»** 20](#_Toc185184779)

[**2.2.1.2.2**. **Модуль «dao»** 21](#_Toc185184780)

[**2.2.1.2.3**. **Модуль «exception»** 22](#_Toc185184781)

[**2.2.1.2.4**. **Класс «ResponseDto»** 2](#_Toc185184782)3

[**2.2.1.2.5**. **Класс «ResponseInterceptor»** 2](#_Toc185184783)3

[**2.2.1.3.** **Модуль «lesson»** 2](#_Toc185184784)5

[**2.2.1.3.1**. **Модуль «bean»** 2](#_Toc185184785)5

[**2.2.1.3.2**. **Модуль «controller»** 26](#_Toc185184786)

[**2.2.1.3.3**. **Модуль «dao»** 27](#_Toc185184787)

[**2.2.1.3.4**. **Модуль «dto»**](#_Toc185184788) 28

[**2.2.1.3.5**. **Модуль «exception»**](#_Toc185184789) 29

[**2.2.1.4.** **Модуль «mail»** 30](#_Toc185184790)

[**2.2.1.4.1**. **Модуль «configuration»** 30](#_Toc185184791)

[**2.2.1.4.2**. **Модуль «dto»** 30](#_Toc185184792)

[**2.2.1.4.3**. **Модуль «service»** 31](#_Toc185184793)

[**2.2.1.5.** **Модуль «user»** 31](#_Toc185184794)

[**2.2.1.5.1**. **Класс «SecurityConfig»** 31](#_Toc185184795)

[**2.2.1.5.2**. **Класс «TestServiceApplication»** 3](#_Toc185184796)3

[**2.2.2. Папка** **«resources»** 33](#_Toc185184797)

[**2.2.2.1. Директория** **«db.migration»** 33](#_Toc185184798)

[**2.2.2.2. Файл «application»** 35](#_Toc185184799)

[**2.3 Описание интерфейса пользователя** 36](#_Toc185184800)

[**2.4. Уязвимости и методы защиты** 38](#_Toc185184801)

[**2.4.1. SQL-инъекции** 38](#_Toc185184802)

[**2.4.2. CORS (Cross-Origin Resource Sharing)** 39](#_Toc185184803)

[**2.5. Выводы и результаты** 40](#_Toc185184804)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 42](#_Toc185184804)

ВВЕДЕНИЕ

Темой курсовой работы является разработка сервисного приложения для курьеров. Для реализации приложения была выбрана следующая архитектура:

A diagram of a diagram

Description automatically generated

Рисунок 1 – Контейнер веб-сайта для онлайн-тестирования по математике

A diagram of a software company

Description automatically generated

Рисунок 2 - Компонент веб-сайта для онлайн-тестирования по математике – Backend

1. ВЫБРАННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ

Java:

* Язык программирования: Java — это объектно-ориентированный язык программирования общего назначения, который широко используется для разработки различных приложений, от мобильных приложений до enterprise-систем.
* Преимущества: кроссплатформенность, обширная экосистема, высокая производительность, безопасность.

Spring Boot:

* Фреймворк: Spring Boot упрощает разработку Spring-приложений за счет автоконфигурации, встроенного сервера приложений и других удобных функций.
* Преимущества: ускоренная разработка, упрощенная конфигурация, легкое развертывание, хорошая интеграция с другими технологиями.

Spring Security:

* Фреймворк**:** Spring Security — это мощный и настраиваемый фреймворк для обеспечения безопасности приложений. Он предоставляет механизмы аутентификации, авторизации, защиты от атак.
* Преимущества**:** высокий уровень безопасности, гибкая конфигурация, интеграция с Spring Boot, широкий набор функций.

JWT (JSON Web Token):

* Стандарт**:** JWT — это открытый стандарт (RFC 7519) для безопасной передачи информации между сторонами в виде JSON-объекта.
* Преимущества**:** компактность, самодостаточность (содержит всю необходимую информацию), возможность проверки подлинности.

JPA (Java Persistence API):

* Спецификация**:** JPA — это спецификация Java для объектно-реляционного отображения (ORM). Она определяет API для сохранения, извлечения и управления данными в реляционных базах данных.
* Преимущества**:** упрощает работу с базами данных, повышает производительность разработки, обеспечивает переносимость между различными базами данных.

Swagger:

* Фреймворк**:** Swagger — это набор инструментов для документирования и тестирования REST API.
* Преимущества**:** автоматическая генерация документации, интерактивный интерфейс для тестирования API, упрощение разработки и использования API.

2.ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**2.1.Структура программы**

Программа, разработанная в этом проекте, включает:

* Папка **ru.test\_service**: корневой пакет проекта, содержащий все основные компоненты приложения.
* authorization: Модуль, отвечающий за авторизацию и аутентификацию пользователей.
  + configuration: Классы конфигурации для JWT, фильтров и т.д.
  + controller: Контроллеры для обработки запросов, связанных с авторизацией (логин, логаут).
  + dto: DTO для передачи данных, связанных с авторизацией.
  + exception: Исключения, связанные с авторизацией.
  + service: Сервисы для валидации токенов и т.д.
* common: Модуль, содержащий общие компоненты и утилиты, используемые в разных частях приложения.
  + configuration: Общая конфигурация приложения, например, для Swagger и ModelMapper.
  + dao: Абстрактные классы и интерфейсы для работы с базой данных.
  + exception: Обработка исключений, включая глобальный обработчик исключений.
* lesson: Модуль, отвечающий за управление уроками.
  + bean: Сервисы для работы с уроками.
  + controller: Контроллеры для обработки запросов, связанных с уроками.
  + dao: Entity, repository и сервисы для работы с базой данных.
  + dto: DTO для передачи данных, связанных с уроками.
  + exception: Исключения, связанные с уроками.
* mail: Модуль для отправки email.
  + configuration: Конфигурация для отправки email.
  + dto: DTO для передачи данных email.
  + service: Сервис для отправки email.
* user: Модуль, отвечающий за управление пользователями.
  + bean: Сервисы для работы с пользователями и паролями.
  + configuration: Конфигурация для восстановления пароля.
  + controller: Контроллеры для обработки запросов, связанных с пользователями и восстановлением аккаунта.
  + dao: Entity, repository и сервисы для работы с базой данных.
  + dto: DTO для передачи данных, связанных с пользователями.
  + exception: Исключения, связанные с пользователями.
* SecurityConfig: Конфигурация Spring Security для CORS.
* TestServiceApplication: Главный класс приложения Spring Boot.

Директория **resources:** содержит все не-Java ресурсы проекта, включая конфигурационные файлы и скрипты для базы данных.

* db.migration: содержит SQL-скрипты для управления схемой базы данных. Скрипты, вероятно, используются с инструментом миграции, таким как Flyway или Liquibase.
  + V1\_\_CreateUserTables.sql: Скрипт для создания таблиц, связанных с пользователями (пользователи, роли, связи между ними).
  + V2\_\_CreateTestTables.sql: Скрипт для создания таблиц, связанных с тестами (уроки, тесты, вопросы, ответы, связи между ними).
  + V3\_\_UpdateUser.sql: Скрипт для обновления таблицы пользователей, например, добавления новых полей.
* application.yml: Главный файл конфигурации приложения Spring Boot

A diagram of a computer program

Description automatically generated

Рисунок 3 - Схемы базы данных в виде ER-диаграммы

**2.2. Описание основного кода**

**2.2.1.** **Папка** **«ru.test\_service»**

**2.2.1.1.** **Модуль «authorization»**

Этот модуль отвечает за авторизацию и аутентификацию пользователей в вашем приложении. Он использует JWT (JSON Web Token) для безопасной передачи информации о пользователе и Spring Security для управления доступом к ресурсам. Структура модуля:

##### **2.2.1.1.1**. **Модуль «configuration»**

Этот модуль отвечает за настройку механизмов авторизации и аутентификации в приложении. Он содержит классы конфигурации для JWT, фильтров и управления часовыми поясами пользователей.

* filter**:** Содержит фильтры, которые перехватывают HTTP-запросы.
* JwtAuthFilter: Фильтр для аутентификации пользователей по JWT.
  + doFilterInternal(): Этот метод вызывается для каждого HTTP-запроса. Он проверяет наличие JWT в заголовке Authorization, валидирует токен и, если он действителен, аутентифицирует пользователя, извлекая информацию из токена и сохраняя ее в контексте безопасности Spring Security.
  + jwtTokenAuthentication(): Этот метод выполняет аутентификацию пользователя на основе JWT. Он извлекает информацию о пользователе из токена, создает объект SpringSecurityPrincipal и устанавливает его в контекст безопасности.
* jwt**:** Содержит классы, связанные с созданием, валидацией и управлением JWT.
* JwtConfigurationProperties: Класс для чтения конфигурации JWT из файла application.yml.
  + private String secret: Хранит секретный ключ, который используется для подписи и проверки JWT.
  + private int expiration: Хранит время жизни токена в секундах.
* JwtSecurityConfig: Класс конфигурации Spring Security, который настраивает использование JWT для аутентификации.
  + securityFilterChain(): Этот метод конфигурирует Spring Security. Он определяет, какие URL-адреса требуют авторизации, отключает ненужные функции Spring Security (basic authentication, form login, CSRF, CORS) и добавляет фильтр JwtAuthFilter.
* UserTimeZoneProperties**:** Класс для хранения информации о часовом поясе пользователя, которая считывается из файла конфигурации (application.yml).
* private Integer timezoneOffsetSeconds: Хранит смещение часового пояса пользователя в секундах относительно UTC.

##### **2.2.1.1.2**. **Модуль «controller»:**

Этот модуль отвечает за обработку HTTP-запросов, связанных с аутентификацией и авторизацией пользователей в приложении.

* AuthorizationController**:** Этот контроллер содержит методы для обработки запросов, связанных с авторизацией.
  + login():
    - @PostMapping(«/login»): Обрабатывает POST-запрос на /login для аутентификации пользователя.
    - Принимает AuthorizationRequestDto (логин и пароль) в теле запроса.
    - Извлекает пользователя из базы данных с помощью UserServiceBean.
    - Проверяет пароль, используя PasswordEncoder.
    - Если аутентификация успешна, генерирует JWT с помощью JwtConfigurationProperties и Jwts.builder().
    - Возвращает JWT в AuthorizationResponseDto с HTTP-статусом OK.
  + logout():
    - @PostMapping(«/logout»): Обрабатывает POST-запрос на /logout для выхода пользователя из системы.
    - Очищает контекст безопасности с помощью SecurityContextHolder.clearContext().
    - Возвращает HTTP-статус OK.

##### **2.2.1.1.3**. **Модуль «dto»:**

Этот модуль содержит классы DTO (Data Transfer Object), которые используются для передачи данных, связанных с авторизацией и аутентификацией пользователей, между различными компонентами приложения.

AuthorizationRequestDto**:**

* DTO, используемый для передачи данных для запроса авторизации (логин и пароль) от клиента к серверу.
* private String login: Хранит логин пользователя.
* private String password: Хранит пароль пользователя.
* private Integer userOffsetSecond = UTC\_ZERO\_OFFSET: Хранит смещение часового пояса пользователя в секундах относительно UTC.

AuthorizationResponseDto**:**

* DTO, используемый для передачи JWT клиенту после успешной авторизации.
* private String token: Хранит JWT.

SpringSecurityPrincipal**:**

* DTO, реализующий интерфейс UserDetails Spring Security, который представляет информацию о пользователе после аутентификации.
  + private final UUID id: Хранит ID пользователя.
  + private final String username: Хранит имя пользователя (логин).
  + private final List<SimpleGrantedAuthority> authorities: Хранит список прав доступа пользователя.
  + private final Integer userOffsetSecond: Хранит смещение часового пояса пользователя в секундах относительно UTC.
* TokenValidationResponse**:**
  + DTO, содержащий результат валидации JWT.
  + private final UUID userId: Хранит ID пользователя, извлеченный из JWT.
  + private final boolean isActive: Указывает, является ли JWT действительным.
  + private final Integer userOffsetSecond: Хранит смещение часового пояса пользователя, извлеченный из JWT.

##### **2.2.1.1.4**. **Модуль «exception»:**

Этот модуль содержит пользовательские исключения, которые используются для обработки ошибок, возникающих в процессе авторизации и аутентификации пользователей.

* AuthenticationFailException:
  + Исключение, которое выбрасывается при неудачной аутентификации пользователя. Это может произойти, например, если пользователь ввел неверный логин или пароль, или если JWT недействителен.
  + extends RuntimeException: Наследуется от RuntimeException, что означает, что это unchecked exception, и его не обязательно обрабатывать в блоке try-catch.

##### **2.2.1.1.5**. **Модуль «service»:**

Этот модуль отвечает за валидацию JWT (JSON Web Token), используемого для аутентификации пользователей. Он реализует шаблон проектирования «Стратегия» (Strategy Pattern) для обеспечения гибкости и возможности расширения в будущем.

* TokenValidationService:
  + Интерфейс, который определяет метод validate(String token) для валидации JWT.
  + Этот интерфейс отделяет логику валидации от конкретной реализации, позволяя легко добавлять новые стратегии валидации в будущем.
* TokenValidator:
  + Класс, который реализует интерфейс TokenValidationService и содержит логику для проверки JWT с помощью библиотеки JJWT.
* validate(String token):
  + Использует Jwts.parser() для создания парсера JWT.
  + Использует verifyWith() для проверки подписи JWT с помощью секретного ключа, который получается из JwtConfigurationProperties.
  + Использует parseSignedClaims() для извлечения claims (данных) из JWT.
  + Извлекает ID пользователя и часовой пояс из claims.
  + Возвращает TokenValidationResponse с результатом валидации.

**2.2.1.2.** **Модуль «common»**

Этот модуль содержит общие компоненты, утилиты и конфигурации, используемые в разных частях приложения. Он играет важную роль в обеспечении модульности, переиспользования кода и единообразия в обработке данных и ошибок. Структура модуля:

##### **2.2.1.2.1**. **Модуль «configuration»**

Этот модуль содержит классы конфигурации, которые предоставляют общие настройки для всего приложения.

* OpenAPIConfig**:**
  + Этот класс конфигурирует Swagger (OpenAPI) для автоматической генерации документации API.
  + openAPI():
    - Создает OpenAPI.
    - Устанавливает базовую информацию о API (название, описание, версия).
    - Добавляет требование безопасности SecurityRequirement для использования Bearer-токенов.
    - Определяет схему безопасности SecurityScheme с типом HTTP и схемой bearer.
    - Устанавливает базовый URL API, используя значение из server.servlet.context-path.
* TestServiceConfig**:**
  + Этот класс предоставляет другие общие объекты для приложения.
  + modelMapper():
    - Создает объект ModelMapper из библиотеки ModelMapper, который используется для маппинга объектов (например, между Entity и DTO).
  + secureRandom():
    - Создает объект SecureRandom для генерации криптографически стойких случайных чисел.

##### **2.2.1.2.2**. **Модуль «dao»**

Этот модуль содержит общие компоненты для работы с базой данных, которые используются в разных частях приложения. Он предоставляет базовые классы и интерфейсы для организации доступа к данным и обеспечения единообразия в работе с persistence layer.

* AbstractRepository**:**
  + Интерфейс, который расширяет JpaRepository и может содержать общие методы для всех репозиториев в приложении.
  + Предназначен для добавления методов, которые могут быть полезны во всех репозиториях, например, для поиска по нескольким критериям или для выполнения сложных запросов.
* AbstractServiceDao**:**
  + Абстрактный класс, который предоставляет базовые методы для работы с репозиториями.
  + protected T repository;: Поле, хранящее ссылку на репозиторий.
  + findById(), save(), deleteById(): Примеры методов, которые могут быть определены в этом классе для выполнения основных операций CRUD (Create, Read, Update, Delete).
  + Наследуется классами LessonServiceDao, RoleServiceDao и UserServiceDao в соответствующих модулях.
* BaseEntity**:**
  + Абстрактный класс, который является базовым классом для всех entity в приложении.
  + Может содержать общие поля, такие как id, createdDate, updatedDate, которые присутствуют во всех таблицах базы данных.
  + Наследуется классами Lesson, Role и User в соответствующих модулях.

##### **2.2.1.2.3**. **Модуль «exception»**

Этот модуль содержит классы для обработки исключений в приложении. Он определяет базовый класс для пользовательских исключений, глобальный обработчик исключений и DTO для передачи информации об ошибках.

* AbstractException**:**
  + Абстрактный класс, который служит базовым классом для всех пользовательских исключений в приложении.
  + getMessage(): Абстрактный метод, который должен быть переопределен в подклассах для возврата сообщения об ошибке.
  + getDescription(): Абстрактный метод, который должен быть переопределен в подклассах для возврата подробного описания ошибки.
* GlobalExceptionHandler**:**
  + Класс, который обрабатывает все исключения типа AbstractException, не обработанные в контроллерах.
  + @RestControllerAdvice: Аннотация, которая указывает, что этот класс является глобальным обработчиком исключений для контроллеров.
  + handleException(AbstractException ex):
    - @ExceptionHandler(AbstractException.class): Аннотация, которая указывает, что этот метод обрабатывает исключения типа AbstractException.
    - Логирует ошибку с помощью log.error().
    - Создает ResponseErrorDto с сообщением и описанием ошибки.
    - Возвращает ResponseEntity с HTTP-статусом OK и ResponseErrorDto в теле ответа.
* NotFoundException**:**
  + Пользовательское исключение, которое выбрасывается, когда не найден какой-либо ресурс (например, пользователь или урок).
  + extends AbstractException: Наследуется от AbstractException.
  + Переопределяет методы getMessage() и getDescription() для возврата информации об ошибке «не найден».
* ResponseErrorDto**:**
  + DTO, используемый для передачи информации об ошибках клиенту.
  + extends ResponseDto: Наследуется от ResponseDto.
  + private String description: Хранит подробное описание ошибки.

**2.2.1.2.4**. **Класс «ResponseDto»**

Этот класс определяет структуру данных для ответов API. Он используется для унификации формата ответов и передачи данных клиенту, включая информацию об ошибках.

* errCode:
  + Строковое поле, содержащее код ошибки.
  + Используется для передачи информации об ошибке клиенту.
  + Может быть null, если запрос обработан успешно.
* content:
  + Поле типа Object, содержащее данные ответа.
  + Может содержать любой тип данных, например, список объектов, примитивные значения или другие DTO.

Конструкторы**:**

* ResponseDto(): Конструктор без аргументов.
* ResponseDto(String errCode): Конструктор, принимающий код ошибки.
* ResponseDto(String errCode, Object content): Конструктор, принимающий код ошибки и данные ответа.

**2.2.1.2.5**. **Класс «ResponseInterceptor»**

Этот класс является компонентом Spring, который перехватывает ответы, возвращаемые контроллерами, и модифицирует тело ответа перед отправкой клиенту. Он реализует интерфейс ResponseBodyAdvice<Object>, который позволяет ему взаимодействовать с телом ответа.

* @ControllerAdvice: Аннотация, которая указывает, что этот класс является компонентом для обработки ответов контроллеров.
* @Configuration: Аннотация, которая указывает, что этот класс является классом конфигурации Spring.
* implements ResponseBodyAdvice<Object>: Интерфейс, который позволяет классу взаимодействовать с телом ответа перед отправкой клиенту.
* supports():
  + Этот метод определяет, должен ли ResponseInterceptor применяться к данному методу контроллера. В данном случае он всегда возвращает true, что означает, что он будет применяться ко всем методам контроллеров.
* beforeBodyWrite():
  + Этот метод вызывается перед тем, как тело ответа будет записано в ответ. Он получает тело ответа, информацию о типе возвращаемого значения, информацию о запросе и ответе.
  + if (body == null) { return null; }: Если тело ответа null, ничего не делается.
  + if (body instanceof ResponseErrorDto) { return body; }: Если тело ответа уже является объектом ResponseErrorDto (ошибка), оно не изменяется.
  + if (body.getClass().getPackageName().contains(«ru.test\_service») || body instanceof Page<?>): Если тело ответа является объектом из пакета ru.test\_service или объектом Page (для пагинации), оно заворачивается в ResponseDto.
  + ResponseDto responseDto = new ResponseDto();: Создается новый объект ResponseDto.
  + responseDto.setContent(body);: Оригинальное тело ответа помещается в поле content объекта ResponseDto.
  + return responseDto;: Возвращается ResponseDto в качестве нового тела ответа.

**2.2.1.3.** **Модуль «lesson»**

Этот модуль отвечает за управление уроками в приложении для онлайн-тестирования по математике. Он предоставляет функциональность для создания, чтения, обновления и удаления уроков, а также обработки связанных с этим ошибок. Структура модуля:

**2.2.1.3.1**. **Модуль «bean»**

Этот модуль содержит сервисный класс LessonServiceBean, который отвечает за бизнес-логику, связанную с управлением уроками в приложении.

* LessonServiceBean: Сервис для управления уроками.
* createLesson():
  + @Transactional: Аннотация, указывающая, что метод выполняется в рамках транзакции базы данных.
  + Принимает CreateLessonDto в качестве входных данных.
  + Проверяет, указано ли название урока и не существует ли уже урок с таким названием.
  + Трансформирует CreateLessonDto в entity Lesson с помощью ModelMapper.
  + Сохраняет Lesson в базе данных с помощью LessonServiceDao.
  + Трансформирует сохраненный Lesson обратно в LessonDto и возвращает его.
* updateLesson():
  + @Transactional: Аннотация, указывающая, что метод выполняется в рамках транзакции базы данных.
  + Принимает LessonDto в качестве входных данных.
  + Находит урок в базе данных по ID.
  + Проверяет, указано ли название урока и не существует ли уже другой урок с таким названием.
  + Обновляет название урока.
  + Сохраняет обновленный Lesson в базе данных.
  + Трансформирует сохраненный Lesson обратно в LessonDto и возвращает его.
* findById():
  + @Transactional: Аннотация, указывающая, что метод выполняется в рамках транзакции базы данных.
  + Находит урок в базе данных по ID.
  + Трансформирует Lesson в LessonDto и возвращает его.
* deleteLesson():
  + @Transactional: Аннотация, указывающая, что метод выполняется в рамках транзакции базы данных.
  + Удаляет урок из базы данных по ID.

**2.2.1.3.2**. **Модуль «controller»**

Этот модуль отвечает за обработку HTTP-запросов, связанных с управлением уроками в приложении. Он предоставляет REST API для создания, чтения, обновления и удаления уроков.

* LessonController**:** Этот контроллер содержит методы для обработки запросов, связанных с уроками.
  + getUserById():
    - @GetMapping({«/{id}»}): Обрабатывает GET-запрос на /lesson/{id} для получения урока по ID.
    - @PathVariable UUID id: Извлекает ID урока из пути запроса.
    - Вызывает метод findById() из LessonServiceBean для получения урока из базы данных.
    - Возвращает LessonDto с HTTP-статусом OK.
  + createLesson():
    - @PostMapping: Обрабатывает POST-запрос на /lesson для создания нового урока.
    - @RequestBody CreateLessonDto user: Принимает данные для создания урока (CreateLessonDto) в теле запроса.
    - Вызывает метод createLesson() из LessonServiceBean для создания урока.
    - Возвращает LessonDto с HTTP-статусом OK.
  + updateLesson():
    - @PutMapping: Обрабатывает PUT-запрос на /lesson для обновления данных урока.
    - @RequestBody LessonDto user: Принимает данные для обновления урока (LessonDto) в теле запроса.
    - Вызывает метод updateLesson() из LessonServiceBean для обновления урока.
    - Возвращает LessonDto с HTTP-статусом OK.
  + deleteLesson():
    - @DeleteMapping({«/{id}»}): Обрабатывает DELETE-запрос на /lesson/{id} для удаления урока по ID.
    - @PathVariable UUID id: Извлекает ID урока из пути запроса.
    - Вызывает метод deleteLesson() из LessonServiceBean для удаления урока.
    - Возвращает HTTP-статус OK.

**2.2.1.3.3**. **Модуль «dao»**

Этот модуль отвечает за доступ к данным, связанным с уроками (lessons), в приложении. Он использует Spring Data JPA для взаимодействия с базой данных.

* entity**:** Содержит entity-класс, представляющий таблицу «урок» в базе данных.
  + Lesson:
    - @Entity: Аннотация, указывающая, что этот класс является JPA entity.
    - @Table(schema = «test», name = «tr\_lesson»): Аннотация, указывающая схему («test») и имя таблицы («tr\_lesson») в базе данных.
    - extends BaseEntity: Класс наследуется от BaseEntity, который, вероятно, содержит общие поля, такие как ID, дата создания и дата обновления.
    - private String name;: Поле, хранящее название урока.
* repository**:** Содержит интерфейс репозитория для доступа к данным уроков.
  + LessonRepository:
    - extends JpaRepository<Lesson, UUID>, AbstractRepository<Lesson>: Интерфейс наследуется от JpaRepository и AbstractRepository, предоставляя стандартные методы CRUD (create, read, update, delete) для entity Lesson и, возможно, дополнительные методы, определенные в AbstractRepository.
    - Lesson findByName(String name);: Метод для поиска урока по названию.
* service**:** Содержит сервис для доступа к данным уроков.
  + LessonServiceDao:
    - extends AbstractServiceDao<Lesson, LessonRepository>: Класс наследуется от AbstractServiceDao, который, вероятно, содержит общие методы для работы с базой данных.
    - @Transactional: Аннотация, указывающая, что методы этого сервиса выполняются в рамках транзакции базы данных.
    - findByName(String name): Метод для поиска урока по названию, использующий LessonRepository.

**2.2.1.3.4**. **Модуль «dto»**

Этот модуль содержит классы DTO (Data Transfer Object), которые используются для передачи данных, связанных с уроками, между различными компонентами приложения.

* CreateLessonDto**:**
  + DTO, используемый для передачи данных, необходимых для создания нового урока, от клиента к серверу.
  + private String name;: Хранит название урока.
* LessonDto**:**
  + DTO, используемый для передачи информации об уроке.
  + private UUID id;: Хранит уникальный идентификатор урока.
  + private String name;: Хранит название урока.

**2.2.1.3.5**. **Модуль «exception»**

Этот модуль содержит пользовательские исключения, которые используются для обработки ошибок, возникающих при работе с уроками.

* LessonNameHasAlreadyExistException:
  + Исключение, которое выбрасывается, когда пользователь пытается создать урок с именем, которое уже существует в системе.
  + extends AbstractException: Наследуется от AbstractException (из пакета ru.test\_service.common.exception), который предоставляет базовый функционал для пользовательских исключений.
  + getMessage(): Возвращает код ошибки «LESSON\_NAME\_HAS\_ALREADY\_EXIST».
  + getDescription(): Возвращает описание ошибки «Урок с указанным названием уже зарегистрирован в системе».
* LessonNameNotSpecifiedException:
  + Исключение, которое выбрасывается, когда пользователь не указывает имя урока при создании нового урока.
  + extends AbstractException: Наследуется от AbstractException.
  + getMessage(): Возвращает код ошибки «LESSON\_NAME\_NOT\_SPECIFIED».
  + getDescription(): Возвращает описание ошибки «Название урока не указано».

**2.2.1.4.** **Модуль «mail»**

Этот модуль предоставляет функциональность для отправки электронных писем из приложения. Он использует Spring Mail для интеграции с SMTP-сервером и отправки сообщений. Структура модуля:

**2.2.1.4.1**. **Модуль «configuration»**

Этот модуль отвечает за чтение и хранение конфигурации для отправки email из файла application.yml или application.properties.

* MailConfiguration:
  + @Configuration: Аннотация, указывающая, что это класс конфигурации Spring.
  + @ConfigurationProperties(prefix = «spring.mail»): Аннотация, указывающая, что этот класс будет считывать свойства конфигурации с префиксом spring.mail.
  + private String username: Хранит имя пользователя (email адрес) для учетной записи, используемой для отправки email. Этот адрес будет использоваться в качестве отправителя писем.

**2.2.1.4.2**. **Модуль «dto»**

Этот модуль содержит DTO (Data Transfer Object), используемый для передачи данных, необходимых для отправки email.

* EmailDto:
  + DTO, содержащий информацию, необходимую для отправки email.
  + private String subject: Хранит тему письма.
  + private String body: Хранит тело письма (может быть HTML).
  + private String emailSender: Хранит адрес электронной почты отправителя.
  + private String recipient: Хранит адрес электронной почты получателя.

**2.2.1.4.3**. **Модуль «service»**

Этот модуль отвечает за отправку электронных писем в приложении, например, для восстановления пароля или отправки уведомлений.

* EmailService:
  + Сервис, который предоставляет метод sendEmail() для отправки электронных писем.
  + sendEmail(EmailDto emailDto):
    - Принимает EmailDto, содержащий информацию об отправителе, получателе, теме и теле письма.
    - Использует JavaMailSender для отправки письма.
    - Создает MimeMessageHelper для удобной работы с MIME-сообщениями.
    - Устанавливает получателя (setTo()), отправителя (setFrom()), тему (setSubject()) и текст письма (setText()), поддерживая HTML-форматирование.
    - Логирует информацию об успешной отправке или ошибке.

**2.2.1.5.** **Модуль «user»**

Этот модуль отвечает за управление пользователями в приложении. Он предоставляет функциональность для создания, чтения, обновления, удаления пользователей, а также для восстановления паролей. Структура модуля:

**2.2.1.5.1**. **Класс «SecurityConfig»**

Этот класс отвечает за конфигурацию безопасности приложения, в частности, за настройку CORS (Cross-Origin Resource Sharing). CORS - это механизм, который позволяет контролировать доступ к ресурсам вашего приложения с других доменов.

* @Configuration: Аннотация, указывающая, что этот класс является классом конфигурации Spring.
* @EnableWebSecurity: Аннотация, включающая Spring Security.
* @Value(«${url}») public String url: Инъекция значения свойства url из файла конфигурации (application.yml или application.properties). Это свойство должно содержать URL-адрес или список URL-адресов, которым разрешено делать кросс-доменные запросы к вашему приложению.
* corsFilter():
  + @Bean: Аннотация, указывающая, что этот метод создает bean-компонент Spring – это класс, который управляется Spring.
  + Метод создает и настраивает CorsFilter:
    - CorsConfiguration config = new CorsConfiguration(): Создает объект конфигурации CORS.
    - config.addAllowedOrigin(url): Добавляет разрешенные источники (origins) из свойства url.
    - config.addAllowedMethod(«\*»): Разрешает все HTTP методы (GET, POST, PUT, DELETE, etc.).
    - config.addAllowedHeader(«\*»): Разрешает все заголовки.
    - config.setAllowCredentials(true): Разрешает отправку cookies и учетных данных.
    - UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource(): Создает источник конфигурации CORS на основе URL.
    - source.registerCorsConfiguration(«/\*\*», config): Применяет конфигурацию CORS ко всем путям (/\*\*).
    - return new CorsFilter(source): Возвращает настроенный фильтр CORS.

**2.2.1.5.2**. **Класс «TestServiceApplication»**

Этот класс расположен в корневом пакете ru.test\_service и является главным классом приложения Spring Boot. Он отвечает за запуск приложения и выполнение начальной конфигурации.

* @SpringBootApplication: Эта аннотация является комбинацией трех аннотаций:
  + @EnableAutoConfiguration: Включает автоконфигурацию Spring Boot, которая автоматически настраивает объекты на основе зависимостей, найденных в classpath.
  + @ComponentScan: Сканирует текущий пакет и его подпакеты на наличие компонентов Spring (контроллеры, сервисы, репозитории и т.д.) и регистрирует их в контексте приложения.
  + @Configuration: Помечает этот класс как класс конфигурации Spring.
* public static void main(String[] args):
  + Метод main является точкой входа в приложение Java.
  + SpringApplication.run(TestServiceApplication.class, args);: Эта строка запускает приложение Spring Boot.
    - TestServiceApplication.class: Указывает класс приложения, который Spring Boot использует для конфигурации.
    - args: Передает аргументы командной строки (если есть) в приложение.

**2.2.2. Папка** **«resources»**

**2.2.2.1. Директория** **«db.migration»**

Этотдиректория содержит SQL-скрипты для управления схемой базы данных. Эти скрипты, вероятно, используются с инструментом миграции базы данных, таким как Flyway или Liquibase. Каждый скрипт представляет собой миграцию, которая вносит изменения в структуру базы данных.

* V1\_\_CreateUserTables.sql:
  + Создает схему users, если она не существует.
  + Создает таблицы, связанные с пользователями:
    - users.tr\_user: Таблица для хранения информации о пользователях (ID, логин, пароль).
    - users.tr\_role: Таблица для хранения информации о ролях пользователей (ID, название, код).
    - users.tr\_user\_role: Таблица для связи пользователей и ролей (many-to-many).
  + Добавляет комментарии к таблицам и столбцам для документирования.
  + Вставляет начальные данные в таблицы tr\_role и tr\_user.
* V2\_\_CreateTestTables.sql:
  + Создает схему test, если она не существует.
  + Создает таблицы, связанные с тестами:
    - test.tr\_lesson: Таблица для хранения информации об уроках (ID, название).
    - test.tr\_test: Таблица для хранения информации о тестах (ID, название, доступ для всех, ID урока).
    - test.tr\_question: Таблица для хранения вопросов теста (ID, текст вопроса, ID теста).
    - test.tr\_answer: Таблица для хранения ответов на вопросы (ID, текст ответа, правильность ответа, ID вопроса).
    - test.tr\_user\_test: Таблица для связи пользователей и тестов.
  + Добавляет комментарии к таблицам и столбцам.
* V3\_\_UpdateUser.sql:
  + Изменяет таблицу users.tr\_user, добавляя новые столбцы:
    - first\_name: Имя пользователя.
    - last\_name: Фамилия пользователя.
    - patronymic: Отчество пользователя.
    - email: Email пользователя.
  + Добавляет комментарии к новым столбцам.

**2.2.2.2. Файл «application»**

Этот файлявляется основным файлом конфигурации приложения Spring Boot. Он содержит настройки для различных аспектов приложения, включая подключение к базе данных, отправку email, конфигурацию сервера, настройки безопасности и другие параметры.

* spring**:**
  + application**:**
    - name: Имя приложения («test-service»).
  + datasource**:** Настройки подключения к базе данных PostgreSQL.
    - url: URL подключения к базе данных (jdbc:postgresql://localhost:5432/test-service).
    - driver-class-name: Имя класса драйвера JDBC для PostgreSQL (org.postgresql.Driver).
    - username: Имя пользователя для подключения к базе данных (postgres).
    - password: Пароль для подключения к базе данных (postgres).
  + mail**:** Настройки для отправки email через SMTP-сервер Mail.ru.
    - host: Адрес SMTP-сервера (smtp.mail.ru).
    - port: Порт SMTP-сервера (465).
    - username: Имя пользователя (email) для аутентификации на SMTP-сервере (amr\_stacenko@mail.ru).
    - password: Пароль для аутентификации на SMTP-сервере.
    - properties: Дополнительные свойства для настройки SMTP.
      * mail.smtp.ssl.enable: Включить SSL.
      * mail.smtp.ssl.protocols: Использовать протокол TLSv1.2.
* server**:** Настройки сервера приложений.
  + port: Порт, на котором будет запущено приложение (9090).
  + servlet.context-path: Контекстный путь приложения (/test-service).
  + ssl.enabled: Отключить SSL.
* url**:** URL для настройки CORS (Cross-Origin Resource Sharing), возможно, для фронтенд-приложения.
* springdoc**:** Настройки Swagger UI для документирования API.
  + swagger-ui.path: Путь к Swagger UI (/swagger-ui.html).
* authentication**:** Настройки аутентификации с использованием JWT.
  + jwt.enabled: Включить JWT.
  + jwt.expiration: Время жизни токена JWT (7200000 секунд).
  + jwt.secret: Секретный ключ для подписи JWT.
* restore-user**:** Настройки для восстановления пароля пользователя.
  + restore-user-username: Email отправителя для писем восстановления пароля.
  + restore-user-subject: Тема письма для восстановления пароля.
  + restore-user-body: Текст письма для восстановления пароля.
* user**:**
  + timezone-offset-seconds: Смещение часового пояса по умолчанию для пользователей (14400 секунд).

**2.3 Описание интерфейса пользователя**

Интерфейс платформы онлайн-тестирования по математике разработан с учетом потребностей следующих категорий пользователей: учеников и учителей.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Компьютерная игра

Автоматически созданное описание

Рисунок 4 – Главная страницы сайта

Учитель имеет возможность вручную добавлять учетные записи для обучающихся, чтобы никто не смог получить доступ со стороны.

Изображение выглядит как снимок экрана, программное обеспечение, текст

Автоматически созданное описание

Рисунок 5 – Личный кабинет пользователя

При регистрации обучающегося указывается его имя, фамилия, логин, пароль и почта, чтобы у обучающегося была возможность восстановить пароль, если он его забыл.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6 – Окно для авторизации пользователя

После входа в учетную запись обучающийся может начать проходить тесты по тем урокам, которые для него доступны. Учитель может создавать как уроки для всех, так и уроки для конкретных обучающихся.

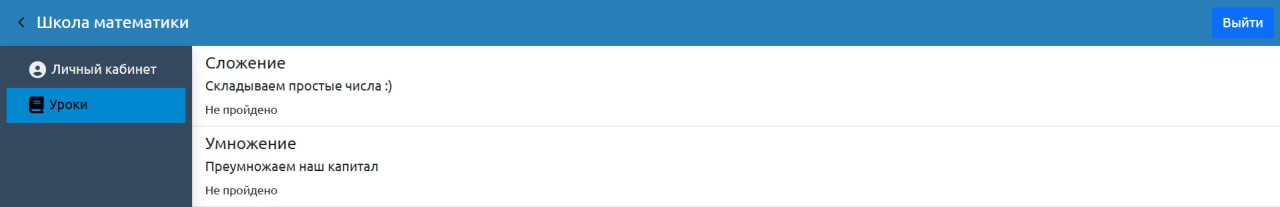


Рисунок 7 – Список доступных уроков

Если кто-то попытается получить список уроков или перейти в личный кабинет без авторизации, то он получит сообщение об ошибке.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 – Сообщение об ошибке при получении уроков

**2.4. Уязвимости и методы защиты**

В данном разделе рассматриваются потенциальные уязвимости приложения "Сайт для онлайн-тестирования по математике для детей" и методы их защиты. Особое внимание уделяется двум распространенным типам уязвимостей: SQL-инъекциям и проблемам, связанным с CORS (Cross-Origin Resource Sharing).

#### **2.4.1. SQL-инъекции**

**Описание:**

SQL-инъекция — это тип атаки, при которой злоумышленник внедряет вредоносный SQL-код в запросы к базе данных. Это может позволить злоумышленнику получить несанкционированный доступ к данным, изменить или удалить данные, а также выполнить другие действия, которые могут нарушить работу приложения и скомпрометировать безопасность данных.

**Методы защиты:**

* **Prepared statements:** Prepared statements — это предварительно скомпилированные SQL-запросы, которые позволяют отделить данные от кода запроса. Это предотвращает внедрение вредоносного кода.
* **Экранирование специальных символов:** Специальные символы в пользовательском вводе, которые могут использоваться для SQL-инъекций, должны быть экранированы или удалены.

**Реализация в проекте:**

В данном проекте для доступа к базе данных используется Spring Data JPA, который по умолчанию использует prepared statements. Это обеспечивает защиту от SQL-инъекций.

Для повышения уровня безопасности и минимизации рисков взаимодействие серверного приложения с базой данных в продакшен-среде будет осуществляться через учетную запись с ограниченными правами. Эта учетная запись будет иметь доступ только для выполнения DML-операций (вставка, обновление, удаление и чтение данных) и не будет обладать правами на выполнение DDL-операций (изменение структуры базы данных).

Таким образом, даже в случае потенциального компрометации приложения или SQL-инъекции злоумышленник не сможет изменить схему базы данных или повлиять на ее структуру.

Для операций, связанных с обновлением схемы (DDL-операции), будет использоваться отдельная учетная запись с повышенными правами, доступ к которой будет строго контролироваться и использоваться только на этапе разработки или при необходимости миграций.

Такой подход обеспечит дополнительный уровень защиты данных и устойчивость системы к потенциальным угрозам безопасности.

#### **2.4.2. CORS (Cross-Origin Resource Sharing)**

**Описание:**

CORS — это механизм безопасности браузера, который ограничивает доступ к ресурсам веб-сайта с других доменов. Если CORS настроен неправильно, злоумышленники могут использовать уязвимости для получения доступа к данным пользователей или выполнения несанкционированных действий.

**Методы защиты:**

* **Правильная конфигурация CORS:** необходимо разрешить доступ только с доверенных доменов и ограничить разрешенные методы HTTP и заголовки.

**Реализация в проекте:**

В данном проекте CORS настроен в классе SecurityConfig. Домен, которому разрешен доступ, указывается в свойстве url в файле application.yml. Важно убедиться, что это свойство содержит только доверенные домены.

**Дополнительные меры безопасности:**

* **Валидация данных:** Валидация всех входных данных от пользователя для предотвращения атак, связанных с некорректными данными.
* **Шифрование данных:** Шифрование конфиденциальных данных, таких как пароли, для защиты от несанкционированного доступа.
* **Регулярное обновление зависимостей:** Обновление библиотек и фреймворков, используемых в проекте, для устранения известных уязвимостей.
* **Тестирование безопасности:** Проведение регулярного тестирования безопасности для выявления и устранения потенциальных уязвимостей.

**2.5. Выводы и результаты**

В результате работы над проектом была разработана программа, представляющая собой веб-сайт для онлайн-тестирования по математике для детей. Программа успешно реализует поставленные задачи и обладает следующими функциональными возможностями:

* Авторизация и аутентификация пользователей:
  + Реализована система регистрации и входа в систему.
  + Используется JWT (JSON Web Token) для безопасной аутентификации.
  + Предусмотрена возможность восстановления пароля.
* Управление уроками:
  + Администраторы могут создавать, редактировать и удалять уроки.
* Создание и проведение тестов:
  + Возможность создания тестов, привязанных к урокам.
  + Возможность прохождения тестов учениками.
* Система управления пользователями:
  + Администраторы могут управлять учетными записями пользователей.
* Отправка email:
  + Программа может отправлять email для восстановления пароля и других уведомлений.

Технологии:

* Java: язык программирования, использованный для разработки приложения.
* Spring Boot: фреймворк, упрощающий разработку Spring-приложений.
* Spring Security: фреймворк для обеспечения безопасности приложения.
* JWT (JSON Web Token): стандарт для аутентификации.
* JPA (Java Persistence API): спецификация для работы с базами данных.
* Swagger: фреймворк для документирования API.
* PostgreSQL: система управления базами данных.

Результаты:

* Функциональный веб-сайт: создан сайт, который позволяет проводить онлайн-тестирование по математике для детей.
* Безопасность**:** реализована система аутентификации и авторизации для защиты данных.
* Удобство использования: сайт имеет интуитивно понятный интерфейс для детей и администраторов.
* Масштабируемость: использование Spring Boot позволяет легко масштабировать приложение в будущем.

Дальнейшее развитие:

* Расширение функциональности: добавление новых типов вопросов, возможности отслеживания прогресса учеников, интерактивных элементов.
* Улучшение интерфейса: создание более привлекательного и адаптивного дизайна.
* Интеграция с другими системами: например, с системами управления обучением (LMS).

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Основы Java: [Электронный ресурс] // Java – Metanit. URL: https://metanit.com/java/tutorial/

2 Основы Spring: [Электронный ресурс] // Java Spring - Habr. URL: https://habr.com/ru/articles/490586/

3 PostgreSQL для быстрого старта: [Электронный ресурс] // PostgreSQL – Skillbox. URL: https://skillbox.ru/media/code/postgresql-vsye-chto-nuzhno-znat-dlya-bystrogo-starta/

4 PostgreSQL Connection: [Электронный ресурс] // PostgreSQL – Jetbrains. URL: https://www.jetbrains.com/help/idea/postgresql.html#connect-to-postgresql-database

5 Руководство по Blazor: [Электронный ресурс] // Blazor – Metanit. URL: https://metanit.com/sharp/blazor/