**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АНАЛИЗА ДАННЫХ И

ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Направление: 09.03.03 – Прикладная информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ДЛЯ ОРГАНИЗАЦИИ ИНДИВИДУАЛЬНЫХ СПОРТИВНЫХ ТРЕНИРОВОЧНЫХ ПРОГРАММ**

**Работа завершена:**

Студент 4 курса

группы 09-852

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Махиянова И.М.

**Работа допущена к защите:**

Научный руководитель

к.т.н., ст. преподаватель кафедры анализа

данных и технологий программирования

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Матренина О.М.

Заведующий кафедрой

к.ф.-м.н.

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бандеров В.В.

Казань-2022

Содержание

[Аннотация 3](#_Toc106029591)

[Введение 5](#_Toc106029593)

[1. Анализ предметной области 7](#_Toc106029594)

[1.1 Анализ особенностей выстраивания тренировочных программ 7](#_Toc106029595)

[1.2 Обзор аналогичных программных систем 8](#_Toc106029596)

[1.3 Разработка технического задания 12](#_Toc106029597)

[1.4 Выбор программных инструментов 14](#_Toc106029598)

[2. Создание приложения 17](#_Toc106029599)

[2.1. Проектирование информационного хранилища данных 17](#_Toc106029600)

[2.2. Разработка программной части информационной системы 19](#_Toc106029601)

[2.3. Разработка внешнего оформления информационной системы, построение связи с серверной частью 26](#_Toc106029602)

[3. Тестирование Web-приложения 32](#_Toc106029603)

[3.1. Тестирование пользовательского интерфейса приложения 32](#_Toc106029604)

[3.2. Тестирование функциональных элементов приложения 37](#_Toc106029605)

[Заключение 40](#_Toc106029606)

[Список использованных источников 43](#_Toc106029607)

# Аннотация

Тема: «Разработка web-приложения для организации индивидуальных спортивных тренировочных программ».

Объем выпускной квалификационной работы составляет 43 страницы, которые содержат 27 рисунков, 16 листингов кода и 23 использованных источников.

Ключевые слова: web-приложение, тренировочные программы, PostgreSQL, Java, Spring Boot, Spring Security, Vue.js.

Цель работы: разработка web-приложения для составления спортивных тренировочных программ.

По результатам работы, пользователь может самостоятельно составлять себе тренировочные программы, выбирать упражнения, задавать необходимые параметры и вести журнал тренировок. Приложение разделено на серверную и клиентскую часть. Серверная часть написана на языке Java на основе фреймворка Spring Boot. Клиентская часть разработано на основе фреймворка Vue.js.

В выпускной квалификационной работе были представлены все этапы разработки web-приложения.

# Abstract

Thesis: «Development of a web application for the organization of individual sports training programs».

The volume of the final qualifying work is 43 pages, which contain 27 figures, 16 code listings and 23 used sources.

Keywords: web application, training programs, PostgreSQL, Java, Spring Boot, Spring Security, Vue.js.

The purpose of the work: development of a web application for compiling sports training programs.

According to the results of the work, the user can create training programs for himself, choose exercises, set the necessary parameters and keep a training log. The application consists of a server and a client parts. The server part is written in Java based on the Spring Boot framework. The client part is developed based on the Vue.js framework.

The final qualifying work introduces all stages of web application development.

# Введение

В век информационных технологий, когда большая часть деятельности автоматизируется и на рабочих местах все больше появляются роботы, у людей становится больше офисной работы, не требующей большой физической активности. В таких условиях формируется так называемый сидячий образ жизни, из-за чего появляются проблемы со здоровьем, развиваются патологии, ухудшается эмоциональный фон, может развиться депрессия и так далее. Чтобы бороться с этой проблемой, активно продвигают в массы здоровый образ жизни. Люди начали усиленно следить за тем, что они потребляют, интересоваться принципами составления здорового рациона, а также обращать внимание на физическую активность в течение дня. И если у большинства жителей поселков, деревень физический труд присутствовал в их повеседневной жизни, то городские жители нашли себе альтернативу: походы в спортзалы, посещения различных тренировок или танцевальных занятий. Но если ходить в спортзал и просто делать какие-то упражнения, не зная правильной техники выполнения и какой-то теоретической базы, есть вероятность не помочь, а, наоборот, нанести вред организму. В таком случае есть два варианта решения проблемы: довериться специалисту и нанять персонального тренера или самому изучить материал и самостоятельно выстраивать тренировочный план. К сожалению, не у всех есть финансовая возможность выбрать первый вариант, так как услуги фитнес-тренера стоят недешево. И в таком случае люди начинают искать какие-то сторонние источники, инструменты, которые смогли бы заменить им тренера.

Цель работы заключается в создании web-приложения, которое помогло бы пользователю без глубоких знаний в сфере фитнеса самостоятельно выстраивать и регулировать тренировочный процесс.

Для достижения заданной цели были выделены следующие задачи:

1. Провести анализ предметной области, изучить принципы выстраивания тренировочного процесса.
2. Разработать требования (ТЗ).
3. Спроектировать архитектуру приложения и выбрать инструменты для разработки.
4. Разработать базу данных. Сформировать базовые сущности и отношения между ними.
5. Разработать серверную часть приложения.
6. Разработать клиентскую часть приложения. Построить связи с серверной частью.
7. Протестировать web-приложение.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованных источников и приложения.

В введении описаны актуальность, цель работы и выделенные задачи.

Первая глава посвящена анализу предметной области. В этой главе разбираются особенности выстраивания тренировочных программ, проводится обзор готовых программных продуктов, выделяются преимущества и недостатки. Далее разрабатываются функциональные и нефункциональные требования к системе и описываются инструмены для разработки.

Во второй главе описывается процесс создания приложения. В нем содержится описание этапа проектирования информационного хранилища, разработка серверной части приложения и пользовательского интерфейса.

Третья глава посвящена этапу тестирования приложения. В ней описываются тесты на пользовательский интерфейс и на функциональные элементы приложения. Тестирование производится как вручную, так и с помощью автоматизированных тестов.

В заключении описываются итоги проделанной работы. В списке источников содержится литература, которая была использована при выполнении ВКР. В приложении содержится фрагмент кода программы.

1. Анализ предметной области
   1. Анализ особенностей выстраивания тренировочных программ

Чтобы самостоятельно выстраивать тренировочный процесс, нужно учитывать несколько важных факторов: ваша цель, индивидуальные особенности организма, распорядок дня, уровень физической подготовки, сколько времени вы можете выделять тренировкам и так далее.

Основные цели тренировок: похудеть, поддерживать форму или набрать мышечную массу. От этого зависит выбор тренировочной программы и упражнений, частота и длительность занятий. По видам физических нагрузок упражнения делятся на анаэробные (силовые упражнения), аэробные (кардиоупражнения) и комплексные (смесь силовых и кардиоупражнений) [3,4].

Для тех, кто собирается худеть, предпочтительнее будет включать в программу больше аэробных упражнений, то есть кардионагрузку. Это могут быть кардиотренировки, быстрая ходьба, занятия бегом, плаванием, активные игры, такие как футбол, баскетбол, волейбол и так далее. Зимой для этого отлично подойдут лыжи, сноуборд. Кардионагрузка хорошо сжигает жир и ускоряет метаболизм, что так же позволяет сжигать больше калорий. Также желательно в программу включать и силовые упражнения, чтобы сохранить мышечную массу и сделать форму более рельефной. Оптимальный план: в неделю чередовать 1-2 силовые и 2-4 аэробные тренировки [8].

Тем, чья цель – поддержание формы, стоит делать упражнения с собственным весом и умеренным отягощением. В основном это базовые упражнения, включающие несколько мышечных групп и суставов. Также лучше включить в программу 1-2 упражнения на каждую группу мышц и таким образом тренировать все тело, а не разделять тренировки по зонам. Такие занятия можно делать и дома с минимальным оборудованием (гири, утяжелители, фитнес-резинки). При этом организм не должен перегружаться, упражнения должны выполняться в умеренном режиме. Цель таких тренировок – это развитие выносливости, мобильности, сохранение мышечной массы и поддержание тела в тонусе. Для этого также подходят йога, пилатес, аэробика, силовые тренировки. В неделю 2-3 тренировки вполне достаточно [11].

Для набора мышечной массы нужно делать упор на базовые упражнения, которые требуют больше усилий и направлены на большее количество мышц, но также нельзя забывать про изолирующие упражнения, включающие только одну группу мышц и требующие соответственно меньше усилий. Важно сосредоточиться на меньшем количестве повторов, но при этом постоянно увеличивать веса – без этого никакого прогресса не будет. Достаточно делать 4-6 повторов с весом 80-85% от одноповторного максимума [8]. Кардио лучше ограничить или вообще исключить.

Для эффективного контроля и управления своим тренировочным процессом необходимо правильно составлять тренировки. Для этого нужно уметь выбирать упражнения в соответствии с группами мышц. При выборе упражнения необходимо определять в зависимости от цели количество повторов и сетов, а также ознакомиться с техникой выполнения. Также нужно постоянно отслеживать график тренировок для регулярности и прогрессирования. По окончании тренировки нужно сделать небольшой отчет о длительности тренировки, потраченных калориях, задействованных мышцах, своих ощущениях, чтобы потом проанализировать данные и скорректировать тренировки, если это необходимо.

* 1. Обзор аналогичных программных систем

Jefit – персональный дневник тренировок с большим количеством функциональных возможностей. При первом же входе пользователь авторизуется и вводит личные данные, что позволяет приложению высчитывать различные меры и рекомендовать подходящие программы. В нем широкая база данных с упражнениями, которые сопровождаются картинками, демонстрирующими наглядно технику выполнения. Также есть подробная письменная инструкция. Приложение позволяет пользователю самому составлять тренировку и изменять любые параметры, вплоть до времени между подходами. Также имеется календарь, где отмечаются дни, в которые были выполнены тренировки, что помогает отслеживать график.

Достоинства Jefit:

* Большой функционал;
* Большая база данных;
* Наглядная демонстрация упражнений;
* Приятная палитра цветов;
* Возможность добавлять личные фотографии для отслеживания прогресса;
* Большое количество информации;
* Практически полная свобода в действиях пользователя: можно и редактировать, и добавлять новую информацию, и менять упражнения, ставить в другом порядке, удалять и т.д.

Недостатки Jefit:

* Слишком насыщенный интерфейс, что может даже испугать пользователя;
* Интуитивно непонятный интерфейс, пользователь легко может запутаться;
* Устаревший дизайн;
* Приложение только на английском языке;
* Некоторые функциональные возможности доступны только платно;
* Содержится лишняя информация.

Данное приложение предоставляет много функциональных возможностей, но иногда это отвлекает пользователя и мешает ему сосредоточиться на главном.

GymPad – приложение, помогающее вести журнал тренировок и дневник питания. Для использования так же необходима авторизация в системе. В приложении есть каталог упражнений, где все упражнения разделены по группам мышц, что очень удобно в использовании. Также при нажатии на упражнение сбоку появляется текстовое описание, инструкция и изображение, показывающее как правильно необходимо выполнять упражнение. В каталоге программ уже есть две готовые программы для ознакомления пользователя. Также имеется отдельный раздел с замерами тела, где можно ввести личные данные по мере их изменений. В разделе со статистикой можно отслеживать регулярность выполнения тренировок и изменения параметров тела.

Достоинства GymPad:

* Большое количество упражнений с подробным описанием;
* Дружелюбный интерфейс с приятной палитрой цветов и понятной структурой;
* Реализован весь необходимый функционал для ведения журнала тренировок, в том числе и конструктор программ, добавление своих упражнений;
* Имеется готовая база данных с продуктами, так что при их добавлении в дневник питания нет необходимости самой указывать КБЖУ (калории, белки, жиры и углеводы), эти параметры автоматически заполняются;
* Есть возможность создания заметок.

Недостатки GymPad:

* При составлении своей тренировки невозможно посмотреть список добавленных упражнений, соответственно редактировать его и удалять упражнения пользователь так же не имеет возможности.

В целом, данное приложение близко к оптимальному, в нем реализован весь минимальный необходимый функционал и практически нет лишней информации. Однако не совсем продуман раздел с собственными программами тренировок, так как непонятно как их можно использовать в дальнейшем и добавлять в журнал.

Fitmus – многофункциональное приложение, в котором пользователь может вести дневник тренировок, питания, тела, здоровья и записей. В данном приложении помимо составления тренировок и ведения журнала, можно найти огромное количество справочной информации по отдельным упражнениям, по мышцам тела. Также предусмотрен раздел со статьями на разные спортивные темы, которые могли бы быть интересны пользователю. Удобно реализованы конструкторы тренировок и упражнений, где расположено кликабельное изображение с группами мышц тела, и при нажатии на одну из групп отображается список соответствующих упражнений. Также сама тренировка отображается очень понятно, структурированно, с сопровождением изображений, что также облегчает восприятие информации пользователем.

Достоинства Fitmus:

* Структурированное изложение информации;
* Множество функциональных возможностей;
* Наличие отдельного раздела «Библиотека» со справочной информацией;
* Наличие подробных видеоинструкций по использованию приложения;
* Интуитивно понятный интерфейс.

Недостатки Fitmus:

* Устаревший дизайн, отсутствие стилей;
* Не совсем полная база данных (в некоторых упражнениях отсутствуют сопроводительные изображения);
* Много информации, в том числе и о пользователе, без которого можно было обойтись;
* Наличие пустого раздела «Тренер».

Приложение Fitmus не просто дневник тренировок, а скорее набор дневников. И некоторым пользователям такой подход может не подойти, так как бывает трудно отслеживать такое количество направлений одновременно. Такие приложения больше подходят для комплексного подхода, а не просто отслеживанию тренировочного процесса.

* 1. Разработка технического задания

Данное web-приложение разрабатывается для обычных людей, не имеющих достаточно знаний в области фитнеса, чтобы помочь им самостоятельно составлять программу тренировок и график в зависимости от их цели и предпочтений. Основные функции приложения:

* Регистрация и авторизация;
* Обзор упражнений и техник выполнения;
* Обзор готовых программ;
* Создание собственной программы тренировки с возможностью выбора упражнений и указывания количества сетов и повторений;
* Ведение журнала тренировок для отслеживания прогресса;
* Пополнение базы данных упражнений и готовых тренировочных программ.

Были выделены 2 роли: пользователь и администратор. В зависимости от роли различаются сценарии поведения пользователя. Для описания функций по ролям было решено составить UML-диаграмму вариантов использования (Use-case диаграмма). Главными составляющими данной диаграммы являются акторы, варианты использования или сценарии поведения и отношения, которыми они связываются. В нашей диаграмме есть два актора – пользователь и администратор. У каждого актора свой набор сценарий поведения, что и отображено на Рисунке 1:

Изображение выглядит как текст, стол, внутренний, изображение

Автоматически созданное описание

Рисунок 1. Use-case диаграмма ролей «пользователь» и «администратор»

По диаграмме видно, что главная функция администратора – это ведение системы, то есть добавление упражнений и тренировок в базу данных. А все основные функциональные возможности приложения, такие как обзор упражнений и тренировок, ведение журнала, составление собственной программы, принадлежат пользователю.

Системные требования для функционирования приложения:

1. Операционная система: Windows 7, 8, 10 (x64), Linux (x64).
2. Процессор: Intel(R) Core(TM) i5-8265U.
3. Оперативная память: 4 GB
4. Браузер: Google Chrome, Yandex.

Требования к производительности:

* Все web-страницы, генерируемые системой, должны загружаться не более чем за 2 секунды после запроса их.

Требования к безопасности:

* Пользователи обязательно авторизуются для входа в систему и для выполнения всех операций.
* Все личные данные, введенные пользователями, должны быть зашифрованы.
* Система должна позволять пользователям просматривать только свои личные данные.

Требования к доступности:

* Система должна быть доступна пользователям в любое время.

Требования к надежности:

* Если соединение между пользователем и системой разрывается, данные не должны быть утеряны, поэтому должно выполняться резервное копирование данных, чтобы их можно было восстановить.
  1. Выбор программных инструментов

Для разработки серверной части web-приложения был выбран объектно-ориентированный язык Java [14], который является одним из самых популярных и универсальных языков программирования. Java используется в разработке мобильных приложений, веб-сервисов, корпоративных приложений. Преимущество Java в том, что этот язык постоянно развивается, расширяются функциональные возможности, добавляются новые инструменты, которые упрощают и оптимизируют разработку. Также она обладает таким свойством как высокая масштабируемость и кроссплатформенность, то есть этот язык позволяет разработчикам постоянно улучшать свой продукт, расширять его возможности, оптимизировать процессы и запускать один раз написанный продукт на разных операционных системах. И еще одним неоспоримым преимуществом Java является наличие различных библиотек и фреймворков, значительно упрощающих создание программных продуктов.

В качестве среды разработки была выбрана интегрированная среда IntelliJ IDEA. Она очень удобна в использовании, помогает в написании кода, эффективна в рефакторинге. Также она предоставляет широкий набор инструментов для разработки и поддерживает популярные фреймворки, такие как Spring [20], Spring Boot [1], Spring Security [12]. К тому же в самой среде можно напрямую подключаться к базе данных, выполнять запросы и просматривать данные.

Архитектура приложения основана на фреймворке Spring MVC [22]. Он обеспечивает архитектуру паттерна Model – View – Controller. В моделях (Model) хранятся данные приложения. Виды (View) отвечают за отображение этих данных в виде HTML-страничек, которые видит пользователь. Контроллеры (Controller) связывают эти два компонента: обрабатывая запросы пользователя, контроллер подбирает соответствующую модель и передает ее в Вид для отображения данных.

Для создания пользовательского интерфейса был использован Vue.js [23] – современный развивающийся JavaScript-фреймворк с открытым исходным кодом. Он так же соответствует паттерну MVC, поэтому при разработке не возникает противоречий. В Vue.js веб-страницы хранятся в виде компонентов, и это позволяет реализовать разделение кода приложения. Таким образом можно избежать повторения кода, выделяя повторно используемые фрагменты в отдельные компоненты и используя их как шаблоны. Также это значительно улучшает читабельность кода. К тому же Vue.js является гибким фреймворком и легко интегрируем с новыми технологиями. Для верстки web-страниц был использован фреймворк Bootstrap [13]. Для написания клиентской части была выбрана среда WebStorm.

В качестве хранилища данных была выбрана СУБД PostgreSQL [6, 16]. Это объектно-реляционная база данных c открытым исходным кодом. Объектно-реляционный подход обеспечивает работу PostgreSQL с данными как в виде записей, связанных друг с другом отношениями (таблицы), так и в виде объектов, их атрибутов, методов и классов. PostgreSQL может работать с большими объемами данных, ограничений у него не имеется. Также преимуществами являются его кроссплатформенность, гибкость и надежность.

Для проверки аутентификации пользователя был использован JSON Web Tokens (JWT) [15]. JWT – это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2. Авторизация на основе JWT

1. Создание приложения
   1. Проектирование информационного хранилища данных

Процесс разработки приложения начинается с проектирования информационного хранилища. В процессе проектирования базы данных была создана ER-диаграмма компонентов системы (Рисунок 3), которая отображает сущности внутри системы и их взаимосвязи. Для создания диаграммы был использован программный продукт StarUML.

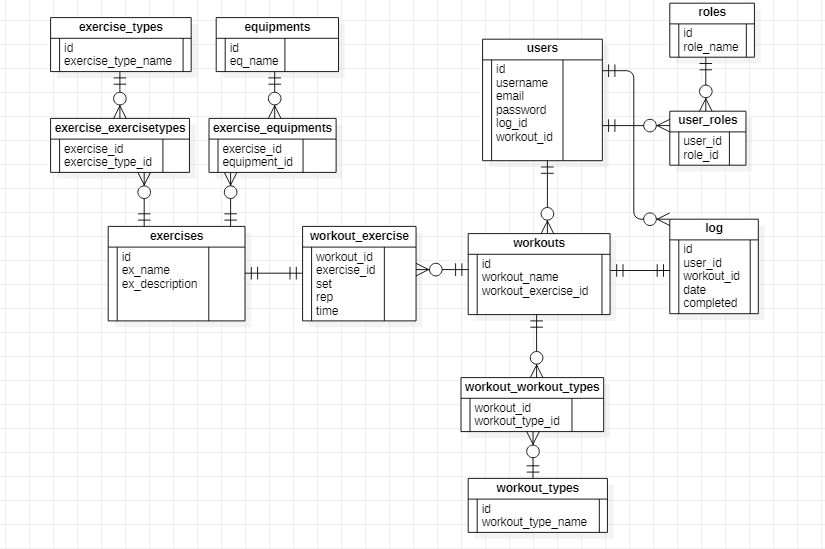


Рисунок 3. ER-диаграмма компонентов системы

В диаграмме отображены следующие основные сущности:

* Users;
* Roles;
* Workouts;
* Log;
* Workout types;
* Workout exercise;
* Exercises;
* Exercise types;
* Equipments.

Users – таблица, в которой хранятся данные о пользователе: username (имя пользователя), email (почта пользователя), password (пароль), log\_id (id записи в журнале – для связи с таблицей log), workout\_id (id тренировки – для связи с таблицей workout). Users связана с таблицей Roles (отвечает за роли пользователей) отношением «многие-ко-многим», поэтому на диаграмме отображена вспомогательная таблица User roles, содержащая в себе id пользователя и id роли. Также Users связана с таблицами Log и Workouts отношением «один-ко-многим».

Таблица Log хранит в себе данные о записях в журнале тренировок: user\_id (id пользователя – для связи с таблицей Users), workout\_id (id тренировки – для связи с таблицей Workouts), date (дата добавления записи), completed (статус тренировки – выполнена/не выполнена).

Таблица Workouts содержит в себе данные о тренировке, такие как: workout\_name (название тренировки), workout\_exercise\_id (id упражнения, включенного в тренировку). Workouts связана с Workout exercise отношением «один-ко-многим» и с Workout types (хранит данные о типах тренировок) отношением «многие-ко-многим», поэтому необходима вспомогательная таблица Workout\_workout\_types, хранящая соответствующие id тренировки и id типа тренировки.

Workout Exercise – таблица, отвечающая за упражнение в конкретной тренировке с такими атрибутами как: workout\_id (id тренировки – для связи с таблицей Workouts), exercise\_id (id упражнения – для связи с Exercises), set (количество подходов), rep (количество повторов), time (длительность упражнения). Этот компонент необходимо было выделить отдельно, потому что в сущности Exercises мы не можем хранить данные о подходах и повторах, так как в зависимости от тренировки и предпочтений пользователя значения могут меняться и чтобы не было повтора одной и той же информации в таблице Exercises мы храним эти данные отдельно.

Таблица Exercises содержит следующую информацию об упражнениях: name (название упражнения), description (описание упражнения). Exercises связана с таблицами Exercise types (тип упражнения) и Equipments (оборудование) отношением «многие-ко-многим», поэтому созданы соответственно две вспомогательные таблицы Exercise\_exercisetypes и Exercise\_equipments.

* 1. Разработка программной части информационной системы

Серверная часть приложения была разработана на основе фреймворка Spring Boot [1], который автоматизирует процедуру настройки и ускоряет процесс создания и развертывания Spring-приложений. **Spring Boot** очень удобен в использовании. У него широкий функционал, также он обладает такими особенностями как управление зависимостями и автоматическая конфигурация. Spring Boot собирает общие зависимости и упаковывает их в так называемые starter-пакеты, что значительно упрощает и ускоряет загрузку необходимых зависимостей. Например, чтобы подключить Spring Security [12] достаточно вписать в файл pom.xml зависимость spring-boot-starter-security и Spring Boot сам автоматически подтянет все необходимые библиотеки. Это позволяет сильно сократить количество строк кода в pom.xml.

Приложение разделено на клиентскую и серверную часть и общение между ними осуществляется в соответствии с принципами REST API [17] с помощью протокола передачи гипертекста HTTP. Действия с объектами осуществляются с помощью REST API методов GET, DELETE, POST, PUT. Их еще называют CRUD-операциями (Create, Read, Update, Delete).

Для взаимодействия с базой данных был использован Spring Data JPA [21]. И в соответствии с таким механизмом были выделены такие компоненты как модели, репозитории и сервисы.

В нашем приложении выделено 10 моделей (Рисунок 4), которые расположены в пакете entity:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4. Классы пакета entity

Данные классы содержат описание сущностей и представляют из себя структуру таблицы в базе данных. Класс помечается аннотацией @Entity, и таким образом Spring [20] понимает, что данный класс является сущностью и при запуске системы автоматически создаст соответствующую таблицу в базе данных. С помощью аннотации @Table мы указываем название таблицы. Также обязательным условием является наличие поля id, который помечается аннотацией @Id и @GeneratedValue(strategy = GenerationType.IDENTITY), что обозначает автоматическую генерацию id. Пример фрагмента класса модели пользователя (Листинг кода №1):

Листинг кода №1. Фрагмент класса User

@Entity  
@Table(name = “users”)  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @NotBlank  
 @Size(max = 20)  
 private String username;  
  
 @NotBlank  
 @Size(max = 50)  
 @Email  
 private String email;  
 @NotBlank  
 @Size(max = 120)  
 private String password;

В данном классе описаны поля id, username, email и password. Также в данном примере реализована валидация данных с помощью аннотаций @NotBlank, проверяющий чтобы поле не было пустым, и @Size, задающий размер поля.

Репозитории являются ключевыми составляющими в Spring Data. Для их реализации был создан пакет repository с набором интерфейсов, соответствующих каждой сущности (Рисунок 5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5. Интерфейсы пакета repository

Классы-репозитории помечаются аннотацией @Repository и наследуются от интерфейса JpaRepository, обеспечивая такие операции как сохранение, извлечение и удаление данных. В качестве примера приведен репозиторий пользователя (Листинг кода №2):

Листинг кода №2. Интерфейс UserRepository

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

Optional<User> findByUsername(String username);

Boolean existsByUsername(String username);

Boolean existsByEmail(String email); }

В данном примере описаны методы findByUsername (поиск по имени пользователя), existsByUsername (проверка на существование пользователя по имени) и existsByEmail (проверка существования пользователя по почте).

Слой сервисов отвечает за всю бизнес-логику приложения. Для каждого класса репозитория создан соответствующий класс сервиса. Все они объединены в пакет service (Рисунок 6):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6. Классы пакета service

Данные классы помечаются аннотацией @Service и в них реализованы такие методы как добавить, получить, удалить или редактировать. Пример метода добавления тренировки в классе WorkoutService (Листинг кода №3):

Листинг кода №3. Метод добавления тренировки в классе WorkoutService

public Workout addWorkout(Long idUser, Workout workout) {

User user = userService.getUserById(idUser);

workout.setUser(user);

return workoutRepository.save(workout); }

Также для реализации паттерна MVC необходимо создать слой Контроллеров, которые отвечают за взаимодействие пользователя с приложением, то есть обработку HTTP-запросов. Классы контроллеров объединены в пакет controller (Рисунок 7):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7. Классы пакета controller

Классы-контроллеры помечаются аннотацией @RestController, который объединяет в себе аннотации @Controller и @ResponseBody. Таким образом у нас создается рест-контроллер в соответствии с REST API, который работает данными, а не моделями. С помощью аннотации @RequestMapping мы связываем класс с URL-адресом. В Листинге кода №4 мы связываем класс WorkoutController с URL-адресом "/api/users/{idUser}/workouts" и говорим, что все методы данного класса относятся к этому адресу.

Листинг кода №4. Фрагмент класса WorkoutController

@RestController

@RequestMapping("/api/users/{idUser}/workouts")

public class WorkoutController {

В контроллерах необходимо описать методы, обрабатывающие запросы пользователей. Чтобы обработать GET-запрос, необходимо пометить метод аннотацией @GetMapping. Также в качестве параметра можно указать URL. В Листинге кода №5 описаны два метода: один возвращает тренировку по id и в таком случае мы указываем в аннотации @GetMapping путь “/{id}”, а другой возвращает список тренировок пользователя по адресу, указанному в аннотации @RequestMapping.

Листинг кода №5. Методы, обрабатывающие GET-запросы, в классе WorkoutController

@GetMapping("/{id}")

public Workout getWorkout(@PathVariable Long idUser, @PathVariable Long id){

return workoutService.getWorkout(idUser, id); }

@GetMapping

public @ResponseBody

Set<Workout> getWorkouts(@PathVariable Long idUser){

return workoutService.getWorkouts(idUser); }

Метод добавления данных (Листинг кода №6) реализует POST-запрос с помощью аннотации PostMapping. Также данный метод помечается аннотацией @ResponseBody, что означает, что на выходе мы автоматически получаем JSON-объект.

Листинг кода №6. Метод добавления тренировки, обрабатывающий POST-запрос

@PostMapping

public @ResponseBody

Workout addWorkout(@PathVariable Long idUser, @Valid @RequestBody Workout workout){

Return workoutService.addWorkout(idUser, workout); }

PUT-запрос обработан в методе редактирования данных и помечается аннотацией @PutMapping (Листинг кода №7).

Листинг кода №7. Метод редактирования тренировки, обрабатывающий PUT-запрос

@PutMapping("/{id}")

public Workout editWorkout(@PathVariable Long idUser, @PathVariable final Long id, @RequestBody Workout workout){

return workoutService.editWorkout(idUser, id, workout); }

Метод удаления данных реализует DELETE-запрос и помечается аннотацией @DeleteMapping. В качестве примера приведен Листинг кода №8 с методом удаления тренировки:

Листинг кода №8. Метод удаления тренировки, обрабатывающий DELETE-запрос

@DeleteMapping("/{id}")

public Workout deleteWorkout(@PathVariable Long idUser, @PathVariable final Long id){

return workoutService.deleteWorkout(idUser, id); }

Отдельно сгруппированы классы, отвечающие за аутентификацию пользователя. Они содержатся в папке security (Рисунок 8):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8. Классы папки security

В классе JwtUtils хранятся данные, необходимые для создания токена: секретный ключ и срок его действия. В этом же классе реализованы методы генерации (Листинг кода №9) и валидации токена и извлечения с помощью него данных о пользователе.

Листинг кода №9. Метод генерации токена

public String generateJwtToken(Authentication authentication) {

UserDetailsImpl userPrincipal = (UserDetailsImpl) authentication.getPrincipal();

return Jwts.builder().setSubject((userPrincipal.getUsername())).setIssuedAt(new Date())

.setExpiration(new Date((new Date()).getTime() + jwtExpirationMs)).signWith(SignatureAlgorithm.HS512, jwtSecret)

.compact(); }

В методе валидации токена реализована проверка на возможные ошибки и вывод соответствующих сообщений (Листинг кода №10).

Листинг кода №10. Метод валидации токена

public boolean validateJwtToken(String authToken) {

try { Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);

return true;

} catch (SignatureException e) {

logger.error("Invalid JWT signature: {}", e.getMessage());

} catch (MalformedJwtException e) {

logger.error("Invalid JWT token: {}", e.getMessage());

} catch (ExpiredJwtException e) {

logger.error("JWT token is expired: {}", e.getMessage());

} catch (UnsupportedJwtException e) {

logger.error("JWT token is unsupported: {}", e.getMessage());

} catch (IllegalArgumentException e) {

logger.error("JWT claims string is empty: {}", e.getMessage()); }

return false; }

В классе AuthTokenFilter происходит проверка запросов на наличие токена. В классе AuthEntryPointJwt обрабатываются ошибки. Также создан отдельный контроллер AuthController, обрабатывающий запросы на авторизацию и регистрацию пользователя. Приложение.

* 1. Разработка внешнего оформления информационной системы, построение связи с серверной частью

Для создания пользовательского интерфейса был использован фреймворк Vue.js .

Vue.js – это прогрессивный JavaScript-фреймворк, пригодный для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию другими библиотеками и существующими проектами.

Интерфейс был разработан в строгом минималистичном стиле преимущественно в темных тонах.

При запуске системы пользователю открывается страничка авторизации (Рисунок 9). Пользователь заполняет поля «логин» и «пароль» и авторизуется в системе.

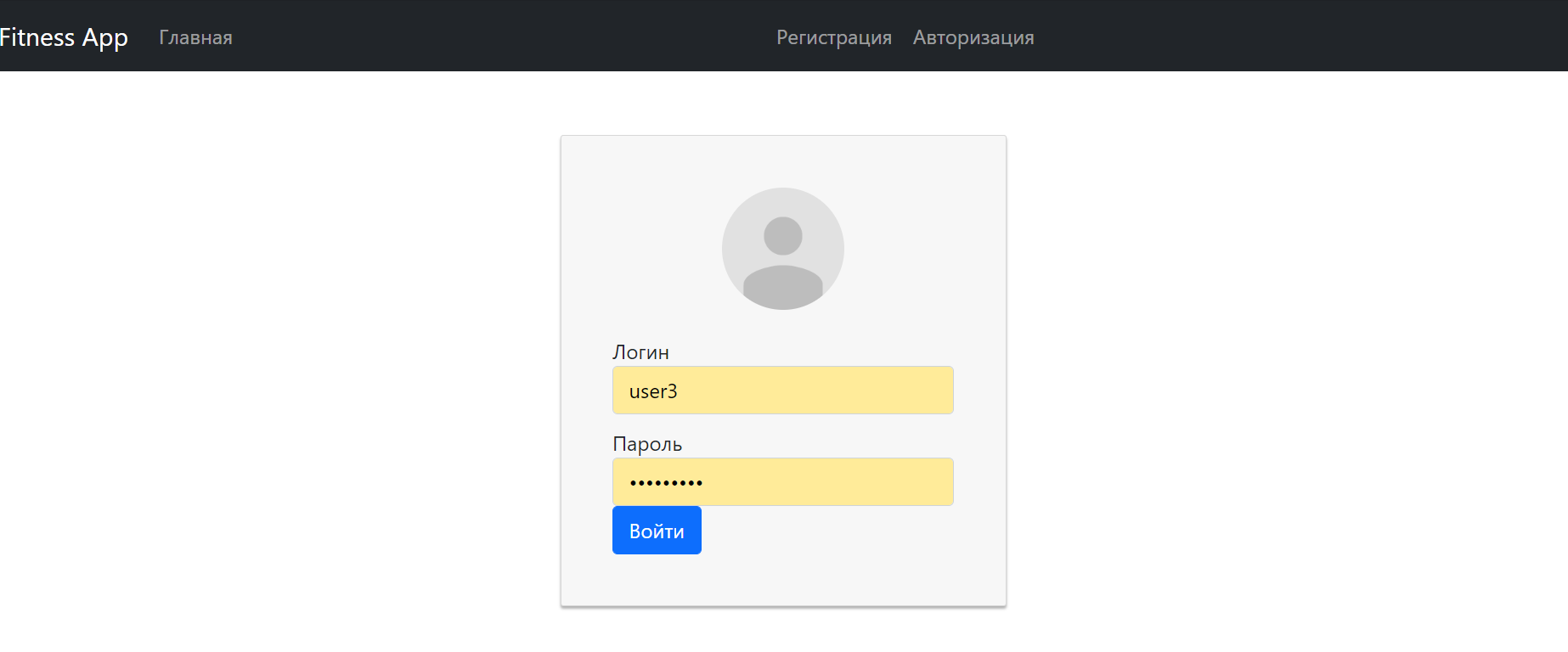


Рисунок 9. Страница авторизации

Если пользователь не зарегистрирован, он может перейти в раздел регистрации и ввести свои данные: логин, e-mail и пароль. При неправильном заполнении полей будет высвечиваться сообщение об ошибке, и система не даст пользователю зарегистрироваться (Рисунок 10).

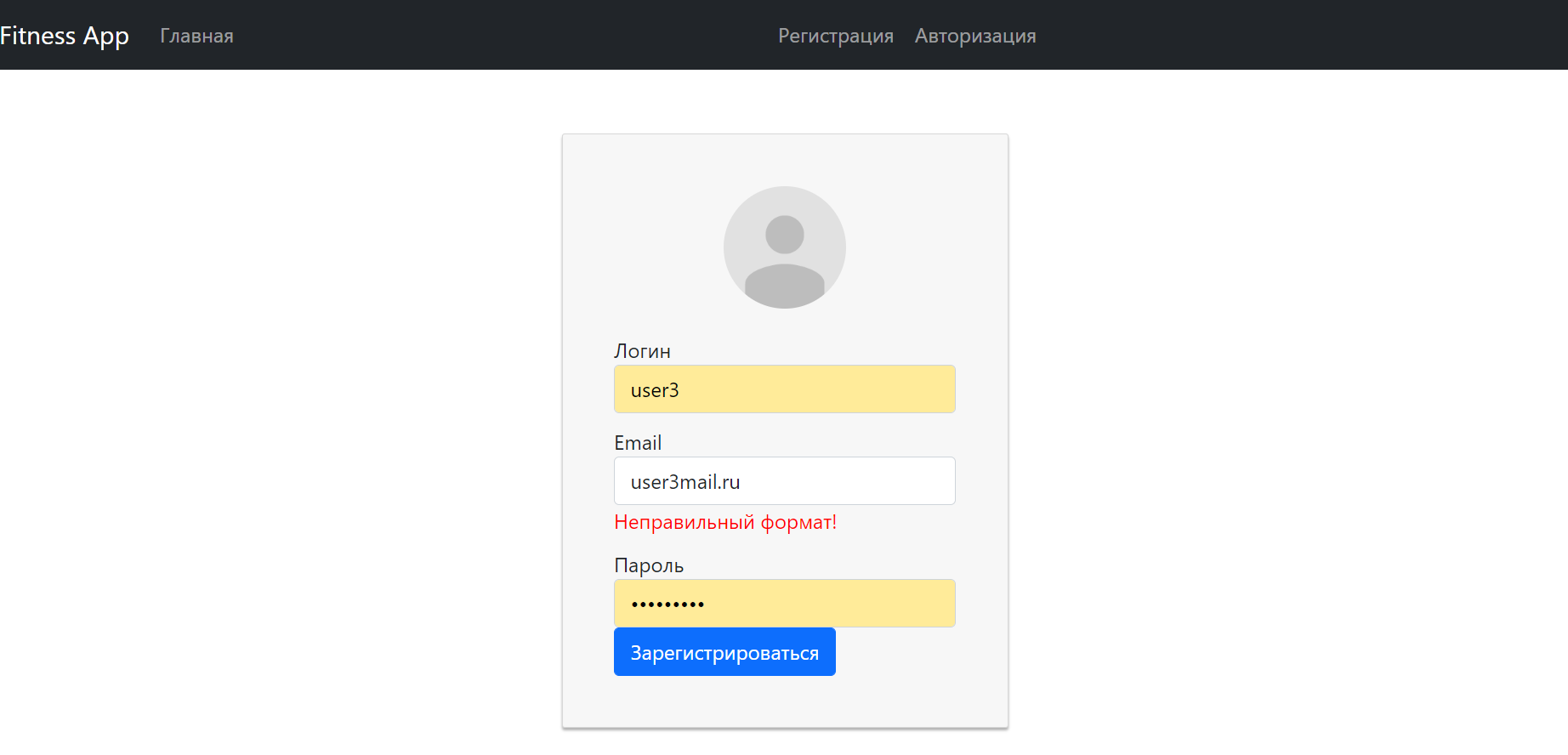


Рисунок 10. Страница регистрации с валидацией полей

В программной части валидация описана в компоненте Register.vue. В приведенном ниже фрагменте заданы размеры поля и вывод сообщений при возникновении ошибок (Листинг кода №11):

Листинг кода №11. Валидация полей при регистрации пользователя

data() {

const schema = yup.object().shape({

username: yup

.string()

.required("Введите логин!")

.min(3, "Минимум 3 символа!")

.max(20, "Максимум 20 символов!"),

email: yup

.string()

.required("Введите email!")

.email("Неправильный формат!")

.max(50, "Максимум 50 символов!"),

password: yup

.string()

.required("Введите пароль!")

.min(6, "Минимум 6 символов!")

.max(40, "Максимум 40 символов!"), });

После того, как пользователь авторизуется, его перебрасывает на Главную страницу (Рисунок 11). На ней отображены карточки «Тренировки», «Упражнения», «Журнал тренировок». При нажатии на кнопку «открыть» будут отображаться соответствующие страницы. Также для перехода на другие страницы можно использовать навигационную панель, расположенную сверху.

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Рисунок 11. Главная страница

На странице «Тренировки» расположены карточки с составленными ранее тренировками (Рисунок 12).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12. Страница «Тренировки»

Также на этой странице можно добавить новую тренировку кликнув на кнопку «Составить тренировку». Тогда откроется новая страничка, где нужно ввести название тренировки и выбрать в выпадающих списках тип тренировки и упражнения (Рисунок 13). И при нажатии на кнопку «Добавить» данные будут сохранены, и новая тренировка отобразится с другими карточками на странице «Тренировки».

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13. Страничка для добавления тренировки

Страничка с упражнениями показана на Рисунке 14. На ней изображена таблица со списком упражнений, их описаниями, типами упражнений и необходимым оборудованием.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14. Страница «Упражнения»

В отличие от обычного пользователя у администратора на страничке с упражнениями есть специальная форма для добавления новых упражнений. Также в таблице добавлена кнопка «Удалить». Страница с упражнениями для администратора изображена на Рисунке 15.

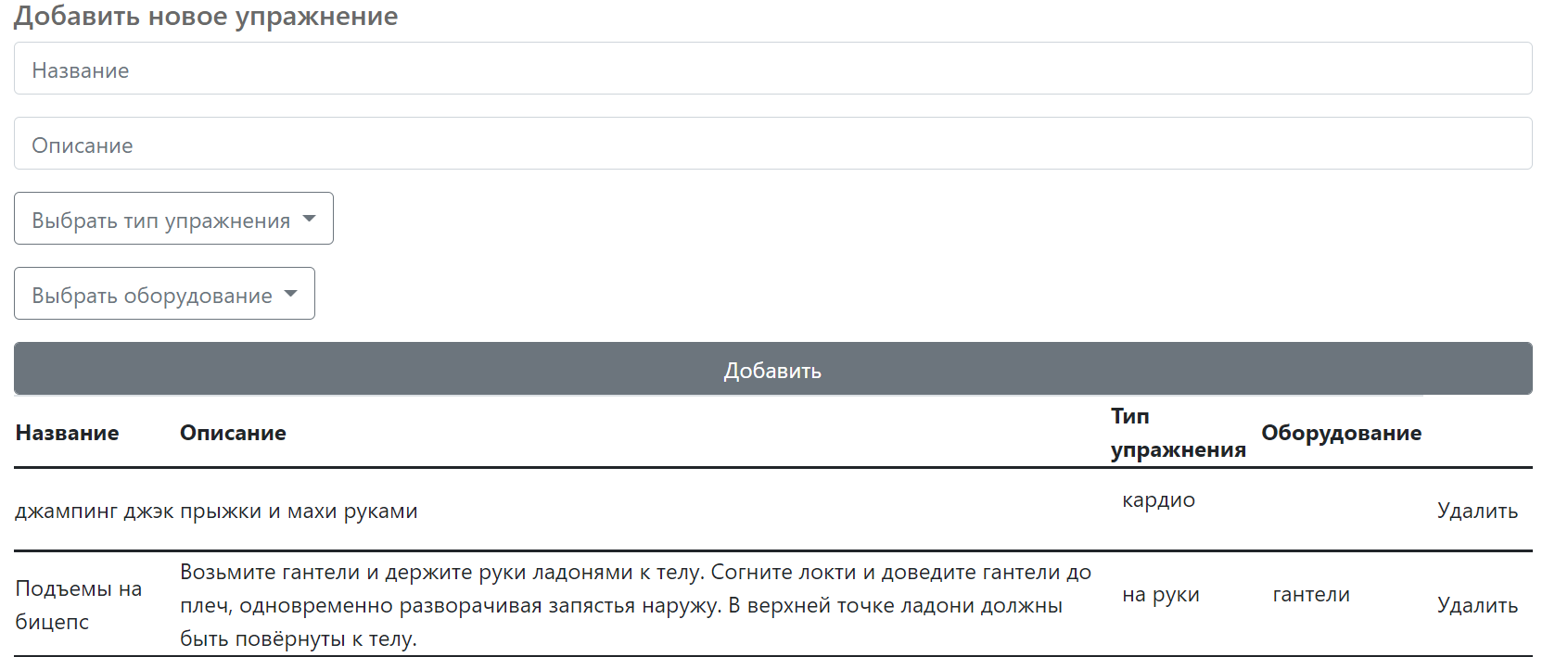


Рисунок 15. Страница «Упражнения» для администратора

Журнал тренировок так же выглядит в форме таблицы с названием тренировки, датой добавления и статусом (Рисунок 16).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 16. Страница «Журнал тренировок

1. Тестирование Web-приложения
   1. Тестирование пользовательского интерфейса приложения

Неотъемлемым этапом разработки web-приложения является тестирование. Оно необходимо для проверки приложения на корректность выполнения запросов, отображений страниц, правильное функционирование элементов, на безопасность и так далее.

Для тестирования приложения были выбраны два метода: автоматизированное тестирование и ручное тестирование. Автоматизированное тестирование очень удобно в использовании, к тому же оно обеспечивает высокую надежность проверки работы приложения и значительно сокращает время написания тестов.

В качестве инструмента для автоматизированных тестов был выбран Selenium IDE [19]. Это плагин, который можно подключить в браузере, и он самостоятельно будет записывать все действия пользователя в системе и автоматически генерировать программный код. Набор действий сохраняются в виде сценариев и могут повторно воспроизводиться для отладки приложения. Тесты проводились в браузере Google Chrome.

Всего было проведено 5 тестов на проверку пользовательского интерфейса:

* Тест на авторизацию пользователя. Рисунок 17.
* Тест на регистрацию и авторизацию нового пользователя. Рисунок 18.
* Тест главной страницы. Рисунок 19-20.
* Тест страницы с тренировками. Рисунок 21.
* Тест добавления тренировки. Рисунок 22-24.

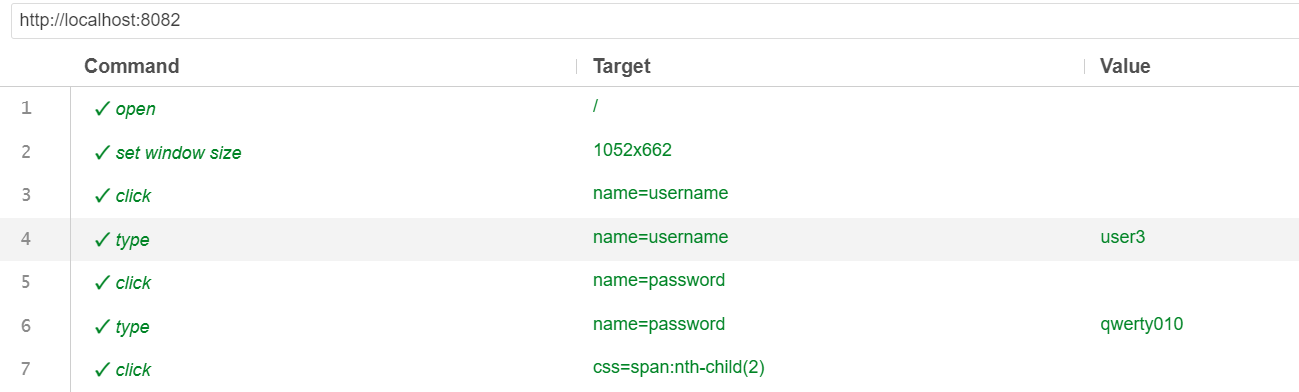
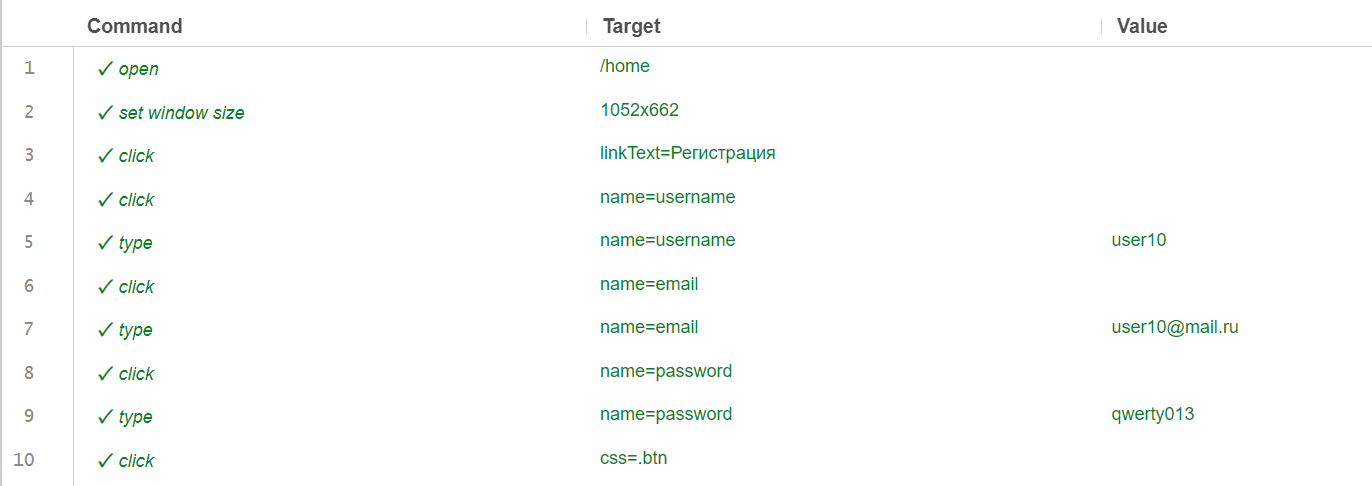


Рисунок 17. Тест на авторизацию пользователя

На Рисунке 17 показано успешное тестирование авторизации пользователя. При запуске теста открывается страница авторизации по адресу http://localhost:8082/login. Далее поля логин и пароль заполняются соответственно значениями «user3» и «qwerty010», после чего нажимается кнопка «Войти» и пользователь входит в систему. Зеленое подсвечивание всех строк означает успешное прохождение теста.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18. Тест на регистрацию и авторизацию нового пользователя

Рисунок 18 демонстрирует успешную регистрацию и авторизацию нового пользователя. При запуске теста открывается страница регистрации по адресу http://localhost:8082/register. После чего автоматически заполняются поля «Логин», «Email» и «Пароль» значениями «user10», «user10@mail.ru» и «qwerty013», нажимается кнопка «Зарегистрироваться», и пользователь успешно регистрируется в системе. Далее осуществляется переход на страницу авторизации и ввод данных, введенных при регистрации, после чего происходит успешная авторизация нового пользователя. Зеленое подсвечивание всех строк означает, что тест успешно пройден.

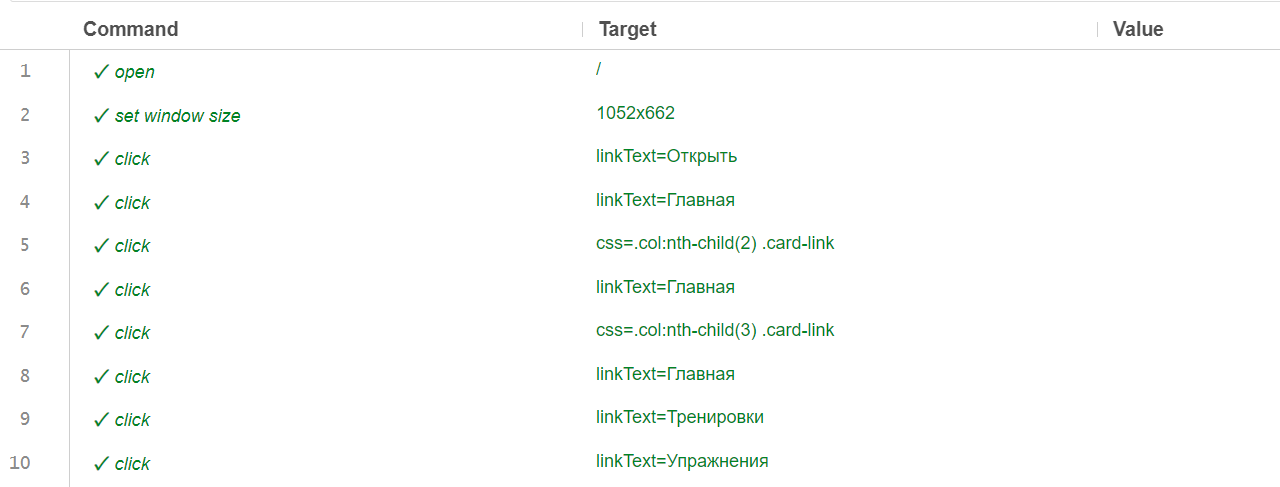


Рисунок 19. Тест главной страницы

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 20. Тест главной страницы

На Рисунках 19–20 показан тест на проверку функционирования главной страницы. В данном тесте проверена корректная работа навигационной панели и кнопок, перенаправляющих на страницы с тренировками, упражнениями и журналом тренировок. Зеленое подсвечивание всех строк означает успешное прохождение теста.

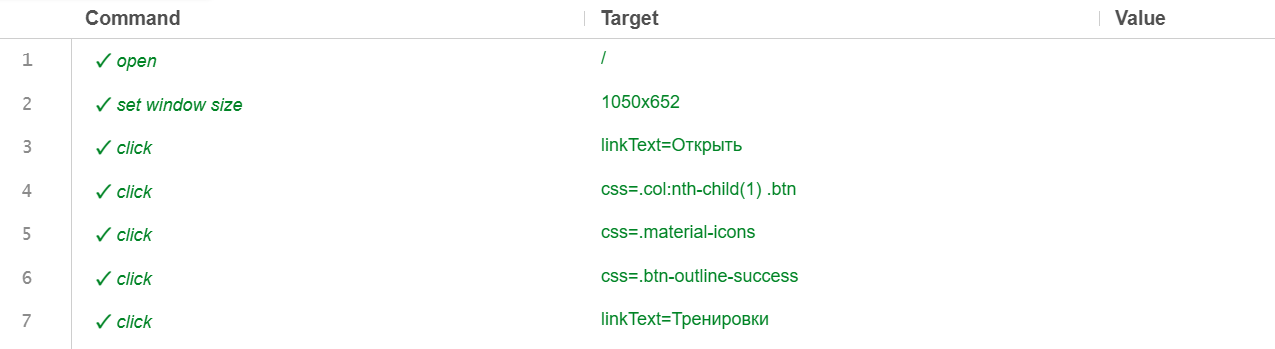


Рисунок 21. Тест страницы с тренировками

Рисунок 21 демонстрирует тест страницы с тренировками. В данном тесте осуществляется проверка открытия и закрытия карточек с тренировками, корректной работы всех элементов страницы. Зеленое подсвечивание всех строк означает успешное прохождение теста.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22. Тест добавления тренировки

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23. Тест добавления тренировки

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24. Тест добавления тренировки

На Рисунках 22-24 показан тест на добавление тренировки. При запуске теста открывается страница с тренировками, нажимается кнопка «Составить тренировку» и открывается отдельная страничка для составления новой тренировки. В ней заполняется поле с названием тренировки, производится выбор типа тренировки и упражнений из выпадающих списков. Далее после нажатия кнопки «Добавить» данные сохраняются, а поля автоматически очищаются. Зеленое подсвечивание всех строк означает успешное прохождение теста.

Также для проверки авторизации пользователя был вручную написан Selenium-тест [5, 18]. Для работы теста был использован Selenium WebDriver.

При написании теста был реализован паттерн Page Object. В классе “LoginPage” прописаны пошагово все методы для проверки: сначала открывается страница с заданным URL-адресом, далее по названию элемента находятся поля с логином и паролем и заполняются отправленными данными, нажимается кнопка с id «auth» и проверяется переход на корректную страницу. Если пользователь с введенными логином и паролем существует в базе данных, следовательно, авторизация проходит успешно и осуществляется переход на главную страницу по адресу http://localhost:8082/home. Ниже приведен метод проверки класса LoginPage (Листинг кода №12):

Листинг кода №12. Метод проверки авторизации в классе LoginPage

public void setMyProject(){

this.driver.get("http://localhost:8082/login"); }

public void enterLogin(String login){

this.driver.findElement(By.name("username")).sendKeys(login); }

public void enterPassword(String password){ this.driver.findElement(By.name("password")).sendKeys(password); }

public void clickBtn(){

this.driver.findElement(By.id("auth")).click(); }

public void check() {

Assert.assertEquals("http://localhost:8082/home", this.driver.getCurrentUrl()); }

Сам тест запускается в классе UITest и в нем мы указываем значения логина и пароля пользователя (Листинг кода №13).

Листинг кода №13. Тест на авторизацию пользователя

@Test

public void SignIn1() throws InterruptedException {

LoginPage loginPage=new LoginPage();

loginPage.setMyProject();

loginPage.enterLogin("user4");

loginPage.enterPassword("qwerty011");

loginPage.clickBtn();

Thread.sleep(10000);

loginPage.check(); }

На Рисунке 25 мы видим, что тест на авторизацию был успешно пройден.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 25. Успешное прохождение теста на авторизацию

* 1. Тестирование функциональных элементов приложения

Функциональное тестирование [10] имеет много различных направлений: регрессионное тестирование, системное тестирование, интеграционное тестирование, тестирование на безопасность и так далее. Мы написали тест для проверки разделов, доступных для пользователя в зависимости от того, какой ролью он обладает: обычный пользователь или администратор.

Для этого теста был создан отдельный контроллер TestController (Листинг кода №14), в котором обрабатываются GET-запросы на проверку роли пользователя.

Листинг кода №14. Класс TestController

@RestController

@RequestMapping

public class TestController {

@GetMapping("/api/user/test")

public String isUser() {

return "simple\_user\_accepted";

}

@GetMapping("/api/admin/test")

public String isAdmin() {

return "admin\_accepted"; }}

В классе SecuredActionsTest мы тестируем этот контроллер с помощью класса MockMvc [9]. Данный класс предназначен именно для тестирования контроллеров без запуска http-сервера. Ниже приведен пример теста для проверки администратора (Листинг кода №15):

Листинг кода №15. Тест на проверку разделов для администратора

@Test

@WithUserDetails("adminTest")

public void testWithGivenRoleAdmin() throws Exception{ mockMvc.perform(MockMvcRequestBuilders.get("/api/admin/test")

.accept(MediaType.ALL))

.andExpect(status().isOk())

.andExpect(content().string(containsString("admin\_accepted"))); }

На Рисунке 26 показаны успешные результаты прохождения тестов:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26. Успешное прохождение тестов на проверку доступных разделов

Также был написан тест, выполняющий поиск пользователя по id. Для проверки отсутствия пользователя в базе данных зададим id несуществующего пользователя и укажем, что ожидаем сообщение со статусом NOT\_FOUND, то есть о том, что пользователь не найден. Ниже приведен код теста (Листинг кода №16):

Листинг кода №16. Тест на поиск пользователя по id

@Test

@WithMockUser

public void getUser() {

ResponseStatusException exception = assertThrows(ResponseStatusException.class, () -> {

userService.getUserById((long) 7); });

HttpStatus expectedMessage = HttpStatus.NOT\_FOUND;

HttpStatus actualMessage = exception.getStatus();

Assert.assertSame(actualMessage, expectedMessage); }

Запустим тест и увидим результаты, сообщающие об успешном прохождении теста (Рисунок 27):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27. Результаты прохождения теста на поиск пользователя по id

Таким образом, на этапе тестирования web-приложения была полностью проверена корректность работы пользовательского интерфейса и было написано несколько тестов для функционального тестирования системы.

# Заключение

По результатам выполнения выпускной квалификационной работы было создано web-приложение для организации индивидуальных спортивных тренировочных программ. В процессе выполнения работы были пройдены все этапы разработки web-приложения.

В ходе анализа предметной области было изучено, какие цели преследуют пользователи при использовании таких приложений и как в зависимости от цели строится тренировочный процесс. Были выделены 3 основные цели: похудение, поддержка веса и набор мышечной массы. Были выделены такие виды упражнений как: анаэробные, аэробные и комплексные. Эти упражнения регулируются в зависимости от цели пользователя.

Также были рассмотрены аналоги приложения. В качестве примеров были взяты 3 web-приложения: Jefit, GymPad, Fitmus. У каждого приложения были свои преимущества и недостатки, которые мы постарались учесть при разработке нашего продукта.

При составлении технического задания в качестве основного стандарта был взят ГОСТ 34.602-89 [2]. В соответствии с ним к приложению были разработаны функциональные и системные требования, а также требования к безопасности, доступности, надежности и производительности. Была составлена UML-диаграмма вариантов использования, наглядно демонстрирующая возможные сценарии поведения пользователя и администратора. В результате выполненной работы все требования были учтены и удовлетворены.

Также были перечислены программные инструменты, использованные при разработке и описана архитектура приложения. Наше приложение имеет клиент-серверную архитектуру и реализует паттерн Spring MVC. Серверная часть была написана на основе фреймворка Spring Boot на языке Java в среде IntelliJ IDEA. Аутентификация пользователя была реализована с помощью Spring Security на основе JWT-токенов. Spring Security – это Java-фреймворк, обеспечивающий безопасную авторизацию и аутентификацию пользователей в системе. Пользовательский интерфейс был написан на основе JavaScript-фреймворка Vue.js в среде WebStorm. Для связи клиентской и серверной части был выбран механизм REST API, который осуществляет общение путем передачи гипертекста HTTP с помощью CRUD-операций. В качестве информационного хранилища была выбрана СУБД PostgreSQL, обеспечивающая объектно-реляционный подход при работе с данными.

Приступая к практической части разработки приложения, первым делом была спроектирована база данных. Для этого была разработана ER-диаграмма, и подробно описаны все сущности и их взаимосвязи. При разработке серверной части для взаимодействия с базой данных был использован Spring Data Jpa. В соответствии с этим механизмом были выделены модели, репозитории и сервисы для каждой сущности из ранее составленной ER-диаграммы. А также были написаны контроллеры, обрабатывающие запросы пользователей в соответствии с архитектурой MVC. Клиентская часть была написана на Vue.js, где каждая страница описана в виде компонента на языке JavaScript.

При проектировании пользовательского интерфейса были выбраны темные нейтральные тона, чтобы пользователь мог сконцентрироваться на нужном и не отвлекаться на внешние составляющие системы. Также был реализован только необходимый функционал, позволяющий составлять тренировочные программы, поэтому приложение не выглядит нагроможденным и не пугает пользователя. Интерфейс довольно прост и интуитивно понятен пользователю, поэтому проблем с использованием возникнуть не должно.

На этапе тестирования приложения были применены 2 метода: автоматизированное и ручное тестирование. Автоматизированное тестирование было выполнено с помощью инструмента Selenium IDE в браузере Google Chrome. Были успешно протестированы все страницы интерфейса, работа функциональных элементов страниц, а также регистрация, авторизация пользователя и добавление данных. С помощью ручного тестирования были проверены доступ пользователя к разделам в зависимости от роли и поиск пользователя по id. Также был написан Selenium-тест проверяющий авторизацию пользователя на основе паттерна Page Object. Все тесты были успешно пройдены, подтвердив корректную работу web-приложения.

Таким образом, поставленная цель работы была достигнута. Было разработано web-приложение, позволяющее пользователю самостоятельно выстраивать и регулировать тренировочный процесс.

В процессе работы были пройдены и успешно выполнены все этапы разработки web-приложения:

* Выполнен анализ предметной области, изучены цели и особенности выстраивания тренировочного процесса;
* Рассмотрены аналоги приложения, выявлены преимущества и недостатки;
* Составлено техническое задание;
* Описаны инструменты для разработки;
* Спроектировано информационное хранилище данных;
* Разработана серверная часть приложения;
* Разработан пользовательский интерфейс приложения. Реализована связь с сервером;
* Протестировано web-приложение.

# Список использованных источников

1. Введение в Spring Boot [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/435144/> (дата обращения: 23.01.2022).
2. ГОСТ 34.602-89 [Электронный ресурс] – Режим доступа: URL: <https://docs.cntd.ru/document/1200006924> (дата обращения: 12.11.2021).
3. Как грамотно составить программу тренировок [Электронный ресурс] – Режим доступа: URL: <https://snow-motion.ru/sostavit-programmu-trenirovok.html> (дата обращения: 10.10.2021).
4. Комплекс упражнений для домашней тренировки [Электронный ресурс] – Режим доступа: URL: <https://www.sports.ru/health/blogs/2836837.html> (дата обращения: 10.10.2021).
5. Нагрузочное тестирование с помощью Selenium [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/168137/> (дата обращения: 11.04.2022).
6. Руководство по PostgreSQL [Электронный ресурс] – Режим доступа: URL: <https://metanit.com/sql/postgresql/> (дата обращения: 10.12.2021).
7. Руководство по Vue.js [Электронный ресурс] – Режим доступа: URL: <https://metanit.com/web/vuejs/> (дата обращения: 14.03.2022).
8. Спортивная энциклопедия SPORTWIKI [Электронный ресурс] – Режим доступа: URL: <http://sportwiki.to/Энциклопедия_бодибилдинга> (дата обращения: 14.10.2021).
9. Тестирование контроллеров с помощью MockMvc [Электронный ресурс] – Режим доступа: URL: <https://sysout.ru/testirovanie-kontrollerov-s-pomoshhyu-mockmvc/> (дата обращения: 15.04.2022).
10. Функциональное тестирование [Электронный ресурс] – Режим доступа: URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 14.04.2022).
11. Эффективные стратегии и программы [Электронный ресурс] – Режим доступа: URL: <https://nabor-massa.ru/kak-sostavit-programmu-trenirovok-samostoyatelno.>html (дата обращения: 13.10.2021).
12. Authentication with Spring Security & JWT [Электронный ресурс] – Режим доступа: URL: <https://www.bezkoder.com/spring-boot-jwt-authentication/> (дата обращения: 26.02.2022).
13. Bootstrap [Электронный ресурс] – Режим доступа: URL: <https://getbootstrap.com/> (дата обращения: 15.04.2022).
14. Java [Электронный ресурс] – Режим доступа: URL: <https://metanit.com/java/> (дата обращения: 10.01.2022).
15. JWT [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/340146/> (дата обращения: 26.02.2022).
16. PostgreSQL [Электронный ресурс] – Режим доступа: URL: https://www.postgresql.org/ (дата обращения: 05.12.2021).
17. REST API [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/company/dataart/blog/277419/> (дата обращения: 28.01.2022).
18. Selenium [Электронный ресурс] – Режим доступа: URL: <https://www.selenium.dev/> (дата обращения: 11.04.2022).
19. Selenium IDE [Электронный ресурс] – Режим доступа: URL: <https://epages.su/blog/kak-avtotesty-uproshchayut-zhizn-selenium-ide-chto-eto-kak-ustanovit-i-ispolzovat.html> (дата обращения: 04.05.2022).
20. Spring [Электронный ресурс] – Режим доступа: URL: <https://spring-projects.ru/> (дата обращения: 21.01.2022).
21. Spring Data JPA [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/435114/> (дата обращения: 02.02.2022).
22. Spring MVC [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/336816/> (дата обращения: 16.01.2022).
23. Vue.js [Электронный ресурс] – Режим доступа: URL: https://ru.vuejs.org/ (дата обращения: 12.03.2022).

Приложение

Методы авторизации и регистрации в AuthController

//авторизация

@PostMapping("/signin")

public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {

Authentication authentication = authenticationManager.authenticate( new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));

SecurityContextHolder.getContext().setAuthentication(authentication);

String jwt = jwtUtils.generateJwtToken(authentication);

UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();

List<String> roles = userDetails.getAuthorities().stream()

.map(item -> item.getAuthority())

.collect(Collectors.toList());

return ResponseEntity.ok(new JwtResponse(jwt, userDetails.getId(), userDetails.getUsername(), userDetails.getEmail(), roles)); }

//регистрация

@PostMapping("/signup")

public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest) { if (userRepository.existsByUsername(signUpRequest.getUsername())) {

return ResponseEntity

.badRequest()

.body(new MessageResponse("Ошибка: Логин занят!")); } if (userRepository.existsByEmail(signUpRequest.getEmail()) { return ResponseEntity

.badRequest()

.body(new MessageResponse("Ошибка: Email занят!")); }

User user = new User(signUpRequest.getUsername(),

signUpRequest.getEmail(),

encoder.encode(signUpRequest.getPassword()));

Set<Role> roles = new HashSet<>();

Role userRoleDefault = roleRepository.findByName(ERole.USER).orElseThrow(() -> new RuntimeException("Ошибка: Роль не найдена."));

roles.add(userRoleDefault);

Role userRoleAdmin = roleRepository.findByName(ERole.ADMIN)

.orElseThrow(() -> new RuntimeException("Ошибка: Роль не найдена."));

roles.add(userRoleAdmin);

user.setRoles(roles);

userRepository.save(user);

return ResponseEntity.ok(new MessageResponse("Пользователь успешно зарегистрирован!")); }