МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Шаблоны программных платформ языка Джава
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: 09.03.04 Программная инженерия

Тема: Приложение «Фирма грузоперевозок»

Студент: Звягинцев Максим Евгеньевич
Группа: ИКБО-16-20
Работа представлена к защите _____ (дата)_____/Звягинцев М.Е. /
(подпись и ф.и.о. студента)

Руководитель: доцент Куликов Александр Анатольевич

Работа допущена к защите_____(дата)_____/Куликов А.А./
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: _____
_____ / _____ /
_____ / _____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень преподавателей, принявших защиту)

М. РТУ МИРЭА. 2022г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Шаблоны программных платформ языка Джава
по профилю: Разработка программных продуктов и проектирование информационных систем
Студент: Звягинцев Максим Евгеньевич
Группа: ИКБО-16-20
Срок представления к защите: _____
Руководитель: доцент Куликов Александр Анатольевич

Тема: Приложение «Фирма грузоперевозок»

Исходные данные: индивидуальное задание на разработку; документация по Spring Framework и JEE, документация по языку Java (версия не ниже 8); инструменты и технологии: JDK (не ниже 8), создание Spring MVC web-приложений, RESTful web-сервисов, Spring ORM и Spring DAO, Gradle, gitHub, IntelliJIDEA. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

Перечень вопросов, подлежащих разработке, и обязательного графического материала:
1. Провести анализ предметной области и формирование основных требований к приложению, 2. Обосновать выбор средств ведения разработки. 3. Разработать приложение с использование фреймворка Spring, выбранной технологии и инструментария. 4.Провести тестирование приложения. 5. Оформить пояснительную записку по курсовой работе 6. Провести анализ текста на антиплагиат 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.
Зав. кафедрой ИиППО: _____/Р. Г. Болбаков/, «_____»_____2022 г.
Задание на КР выдал: _____/А. А. Куликов/, «_____»_____2022 г.
Задание на КР получил: _____/М. Е. Звягинцев/, «_____»_____2022 г.

АННОТАЦИЯ

Курсовая работа по разработке приложения «Фирма грузоперевозок».

Отчет о выполнении курсовой работы содержит 38 страниц, 28 иллюстраций, 5 использованных источников и 1 приложение.

Введение включает в себя обоснование актуальности выбранной темы, цель работы и задачи по её достижению, а также начальную информационную базу для работы с языками и технологиями разработки интернет-ресурса.

Основная часть включает в себя 3 раздела, каждый из которых содержит в себе материал для достижения поставленной цели.

Раздел «Общие сведения» описывает наименование и соответствующую иконку интернет-ресурса, языки, технологии и программное обеспечение, использующиеся в разработке и отладке.

Раздел «Функциональное назначение» описывает назначение веб-страниц приложения.

Раздел «Описание логической структуры» содержит в себе описание структуры проекта веб-приложения, клиентской и серверной частей, межстраничной навигации, оптимизации веб-страниц для различных браузеров и видов устройств.

Раздел «Тестирование» включает в себя постановку задач тестирование и описание их выполнения.

В заключении описывается результат выполнения курсовой работы, выполненные задачи, а также подводятся итоги достижения цели.

Приложение включает в себя файл настройки системы сборки Gradle и файл свойств веб-приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В наши дни жизнь современного человека практически невозможно представить без интернета, благодаря которому почти в любой точке мира всего за несколько секунд любой желающий может получить доступ ко всей открытой информации, а так же воспользоваться бесчисленным множеством различных веб-приложений, которые предлагают абсолютно разнообразный функционал для пользователя.

Таким же образом люди могут обмениваться различной информацией с другими людьми, но нередко возникает необходимость передачи каких-то конкретных вещей, которые выходят за рамки цифрового мира. Можно, конечно, потратить собственное время и деньги и передать необходимую вещь из рук в руки, но, что, если нет возможности сделать это самостоятельно или это финансово невыгодно?

Именно потребность большого количества людей в отправке различных вещей другим людям с помощью удобного сервиса, который всегда будет под рукой в виде веб-приложения, и обосновывает актуальность выбранной темы.

Целью курсовой работы является разработка, тестирование и отладка практичного веб-приложения на тему «Фирма грузоперевозок» с применением различных модулей фреймворка Spring, организацией межстраничной навигации, оптимизацией веб-страниц и размещаемого контента для различных браузеров и видов устройств.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить работу с необходимыми частями фреймворка Spring и СУБД PostgreSQL;
2. Разработать веб-приложение, используя полученные знания;
3. Разработать клиентскую часть веб-приложения;

4. Провести оптимизацию веб-страниц и размещаемого контента для различных браузеров и видов устройств;
5. Провести различные тестирования клиентской и серверной частей веб-приложения.

Информационной базой для выполнения данной работы служат навыки, полученные в течение курса «Шаблоны программных платформ языка Джава», а также интернет-ресурсы «Spring» [1], «Baeldung» [2], «Stack Overflow» [3], «Bootstrap» [4].

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Анализ предметной области

Анализ предметной области данной работы заключается в исследовании существующих веб-приложений, которые предоставляют схожую тематику. В данной части будут рассмотрены примеры сайтов, предоставляющих различные услуги грузоперевозок.

Хорошим примером веб-приложения является сайт компании «UPS» (Рисунок 1.1.1). Сайт предоставляет большое количество разнообразных услуг, а также включает в себя быструю связь для поддержки клиентов и личный кабинет. Стоит отметить и практичный, но не лишенный визуального наполнения дизайн, который привлекает клиентов, но не дает им потеряться в большом количестве контента. Также, немалую роль играет и понятное и выраженное навигационное меню.

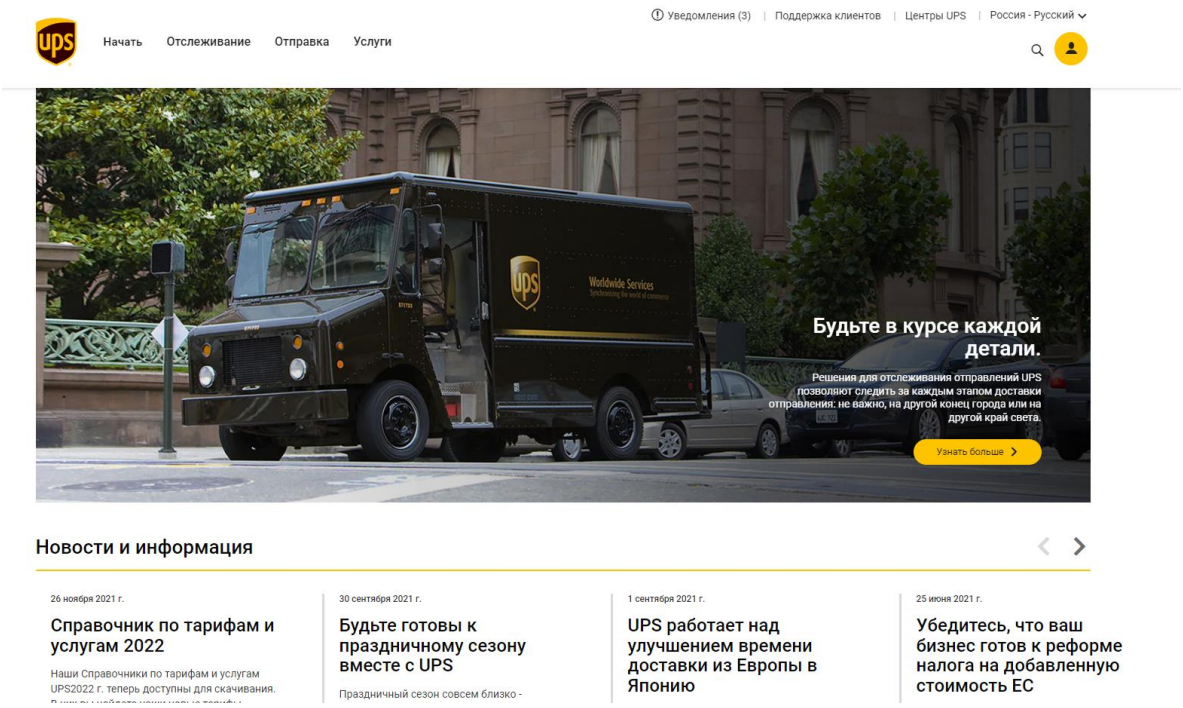


Рисунок 1.1.1 – Сайт компании «UPS»

Не таким хорошим примером можно назвать сайт компании «DHL» (Рисунок 1.1.2), который предлагает даже больший набор предоставляемых услуг, что влечет за собой схожесть хорошей серверной части с веб-приложением компании «UPS» из предыдущего примера. Однако, отсутствие личного кабинета и слишком яркий дизайн, который привлекает по началу, но начинает лишь мешать пользователю в дальнейшем, ставит сайт компании «DHL» на последнюю строчку рейтинга.

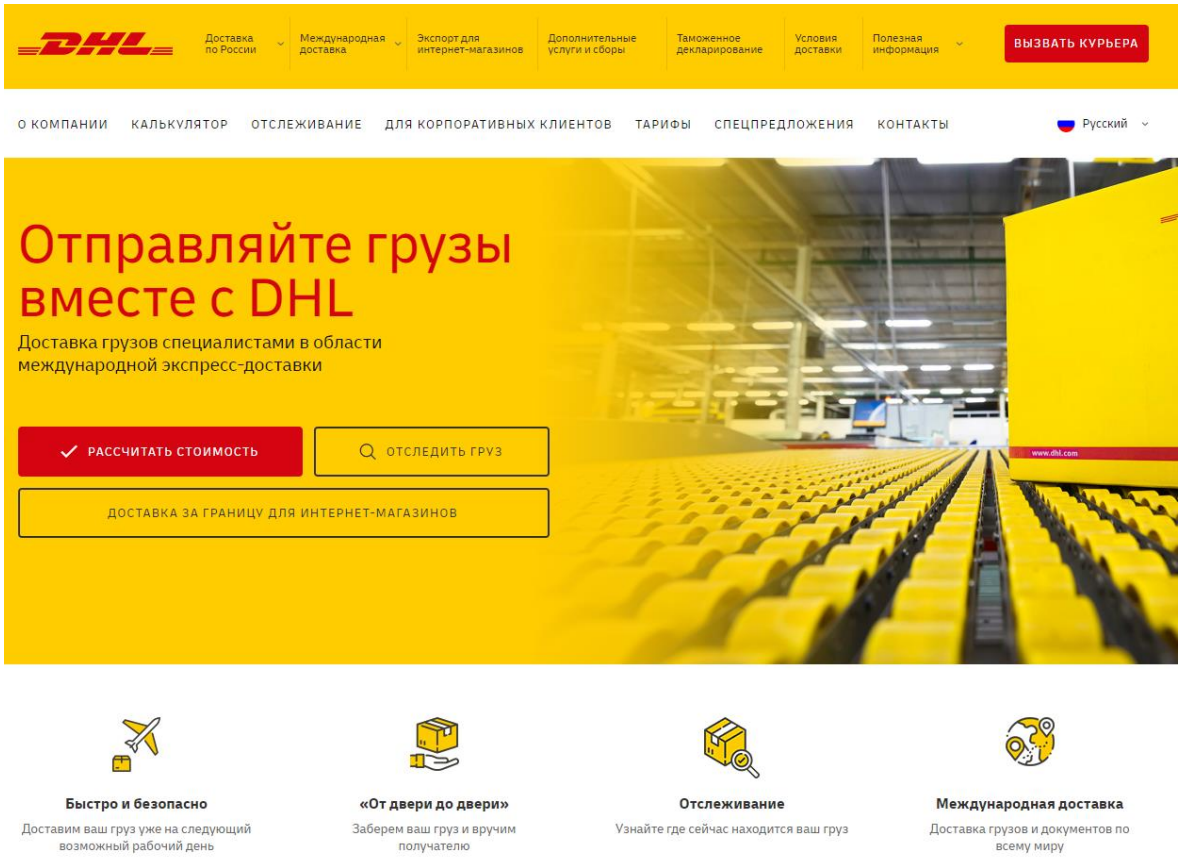


Рисунок 1.1.2 – Сайт компании «DHL»

Фаворитом среди трех рассмотренных веб-приложений компаний, безусловно, является сайт компании «FedEx» (Рисунок 1.1.3), который объединил в меру насыщенный дизайн сайта компании «UPS» и большой набор услуг компании «DHL». Стоит отметить, что сайт этой компании выполнен в приятном и, главное, понятном для пользователя стиле, тем самым, не перегружая видимую область большим количеством информации. Наоборот,

сайт старается распределять свой функционал по определенным категориям, тем самым уделяя большое внимание удобству пользователя.

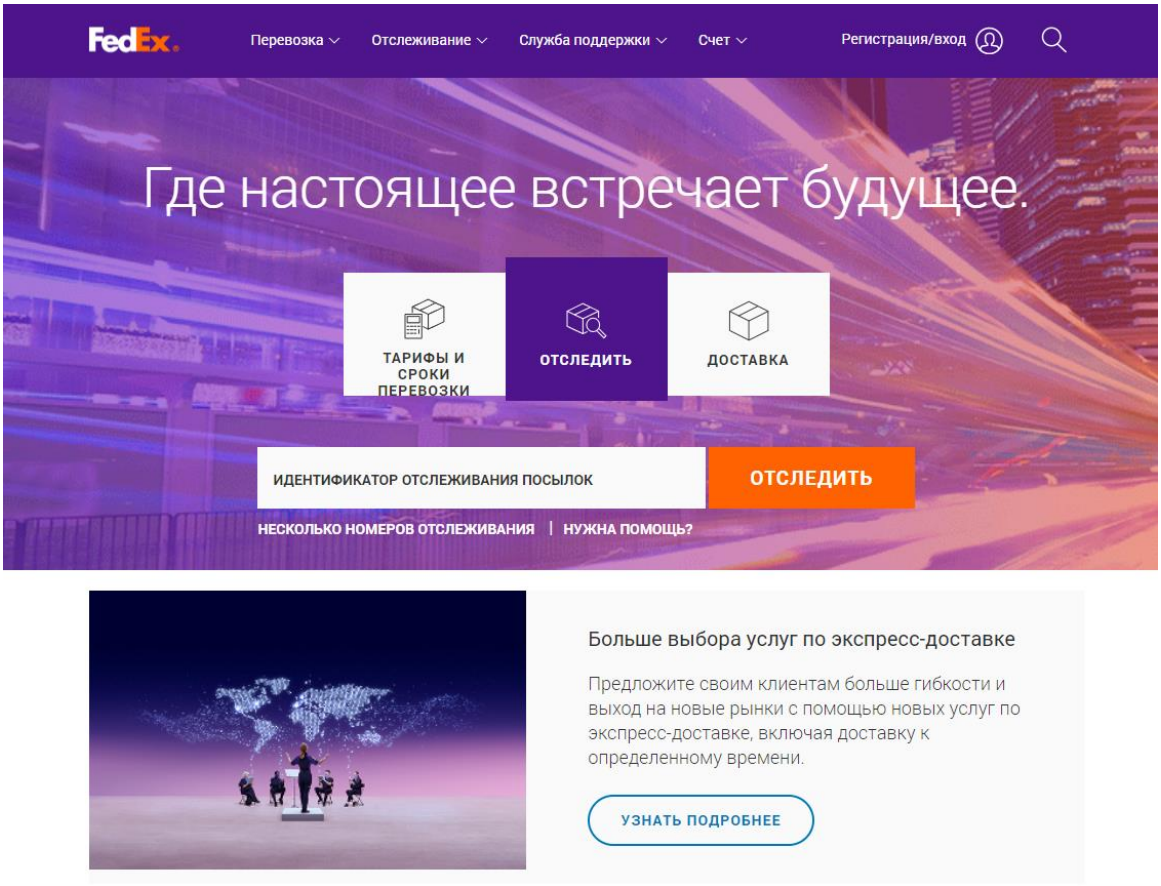


Рисунок 1.1.3 – Сайт компании «FedEx»

Также стоит отметить, что все сайты рассмотренных веб-приложений трех компаний имеют оптимизацию под узкоформатные устройства.

Таким образом, в ходе анализа предметной области, были выявлены различные положительные и отрицательные качества различных существующие веб-приложений, имеющих схожую тематику, в связи с чем были сформированы следующие требования к будущему веб-приложению:

- Практичный и визуально приятный дизайн;
- Наличие навигационного меню;
- Наличие личного кабинета пользователя;
- Наличие понятного и удобного функционала;

- Оптимизация под различные типы устройств.

Отсюда следует, что для реализации будущего веб-приложения необходимо выполнить следующие основные шаги:

1. Реализовать регистрацию и авторизацию пользователей, и сохранение в базе данных;
2. Реализовать личный кабинет пользователей;
3. Реализовать необходимый функционал;
4. Создать визуальную реализацию;
5. Создать навигационное меню;
6. Произвести оптимизацию для различных устройств.

1.2. Обозначение и наименование веб-приложения

Наименованием веб-приложения является «Elephant Express», которое емко описывает его назначение и соответствует выбранной цветовой теме. Так же веб-приложение имеет соответствующую названию иконку (Рисунок 1.2.1), которая была взята с бесплатного сервиса «ICONS8» [5].

Рисунок 1.2.1 – Наименование и иконка веб-приложения

1.3. Прикладное программное обеспечение, необходимое для разработки и функционирования веб-приложения

Для разработки и функционирования интернет-ресурса использовалась интегрированная среда разработки программного обеспечения IntelliJ IDEA (Рисунок 1.3.1) и репозиторий GitHub для контроля версий.

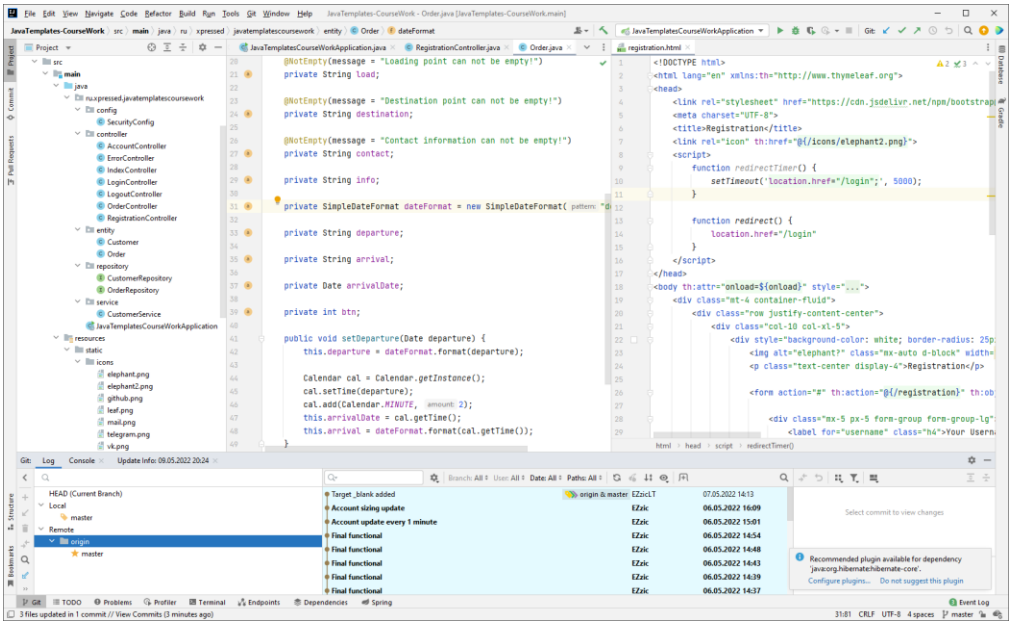


Рисунок 1.3.1 – IntelliJ IDEA

1.4. Языки и технологии реализации веб-приложения

Веб-приложения было реализовано с помощью строго типизированного объектно-ориентированного языка программирования Java 18 и фреймворка Spring, которые использовались для разработки серверной части.

Также для реализации базы данных использовалась свободная объектно-реляционная система управления PostgreSQL, которая существует в различных реализация для множества UNIX-подобных платформ, а также для Microsoft Windows.

В ходе разработки были использованы следующие части фреймворка Spring:

1. Spring Security – модуль, необходимый для построения систем регистрации, аутентификации и авторизации, а также другие возможности для обеспечения безопасности веб-приложения.

2. Spring ORM (Object Relational Mapping) и Spring DAO (Data Access Object) – модули, включающие в себя Spring Data JPA (Java Persistence API), которые необходимы для реализации и работы с базой данных.
3. Spring Thymeleaf – модуль, необходимый для работы с клиентской частью на стороне сервера.

Отдельные технологии, которые использовались во время разработки:

1. Gradle – система автоматической сборки, построенная на принципах Apache Maven, но представленная на языках Groovy и Kotlin вместо традиционной XML-образной формы.
2. Lombok – плагин компилятора, который добавляет в Java новые ключевые слова и превращает аннотации в Java-код, уменьшая усилия на разработку и обеспечивая некоторую дополнительную функциональность.
3. Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений, который включает в себя HTML (HyperText Markup Language) и CSS (Cascade Style Sheet) шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и других компонентов веб-интерфейса.

Выбор технологий для разработки интернет-ресурса обуславливается необходимостью, практичностью, универсальностью и индивидуальными навыками.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Данное веб-приложение имеет сугубо практичный характер, но несмотря на это не лишено небольшого визуального наполнения в виде картинок (Рисунок 2.1.1) и поясняющих иконок, которые были взяты с бесплатного сервиса «ICONS8» [5].

Рисунок 2.1.1 – Визуальное наполнение интернет-ресурса

Веб-приложение оформлено в определенной цветовой теме и стиле, которые были специально подобраны для увеличение визуальной привлекательности, но не лишающие приложение должной строгости для соблюдения необходимой практичности и удобства пользователя.

Каждая веб-страница приложения носит определенный смысл:

- Home (Домашняя страница) – содержит короткое описание веб-приложения.
- Registration (Страница регистрации) – необходима для регистрации новых пользователей в базе данных и последующей авторизации.
- Login (Страница авторизации) – необходима для авторизации зарегистрированных в базе данных пользователей для получения доступа к основным функциям веб-приложения.
- Order (Страница составления заказа) – необходима для оформления заказа на курьерскую или почтовую доставку.

- Account (Страница пользователя) – содержит список заказов, которые были оформлены пользователем, их данные и статус.
- Error (Страница ошибки) – содержит информацию для пользователя о случае ошибки, например, для неправильно введенной ссылки.

Все веб-страницы приложения были условно разделены на две категории: основную и дополнительную.

В основную категорию входят веб-страницы, имеющие первостепенное функциональное назначение для пользователя. К этой категории относятся следующие страницы: Home, Order, Account.

В дополнительную категорию входят веб-страницы, которыми предполагаемый пользователь будет пользоваться редко и в течении непродолжительного срока. К этой категории относятся все остальные страницы, которые не включены в основную категорию.

Отличительной особенностью основных страниц является навигационное меню, с помощью которого пользователь может с легкостью переключаться между ними и всегда понимать, на какой конкретно странице он находится.

Также, каждая основная веб-страница имеет внизу небольшой блок контактной информации (Рисунок 2.1.2), решение о предоставлении которой должно положительно повлиять на удобство связи с автором и предотвращение неправомерного использования интеллектуальной собственности.

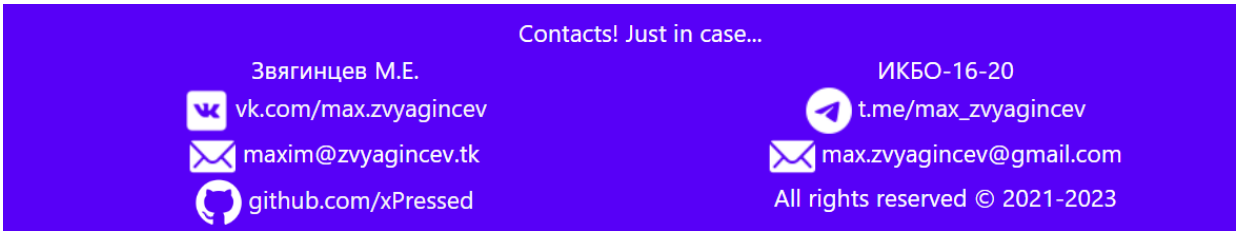


Рисунок 2.1.2 – Контактная информация

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Логическая структура веб-приложения разделяется на две основные части: клиентскую и серверную, каждая из которых выполняет отдельную роль, поэтому они разделены в структуре проекта, реализующего веб-приложение (Рисунок 3.0.1).

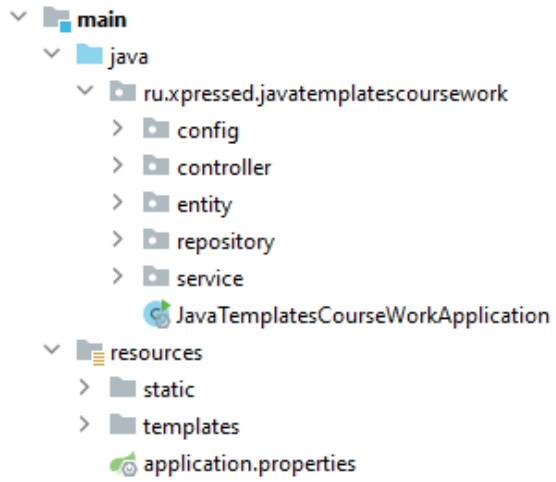


Рисунок 3.0.1 – Логическая структура проекта

3.1. Структура клиентской части веб-приложения

3.1.1. Описание клиентской части

Клиентская часть веб-приложения расположена в директории «resources» в логической структуре проекта и разделяется на несколько поддиректорий:

- Поддиректория «static» используется для хранения необходимым медиа-ресурсов для работы веб-приложения.
- Поддиректория «templates» используется для хранения веб-страниц в формате «*.html».

Все веб-страницы приложения были реализованы с помощью набора инструментов Bootstrap, который использует классы объектов для описания разметки, свойств, типов и других параметров (Рисунок 3.1.1.1).

```
<p class="h3 font-weight-normal mb-5 mt-3 text-center" style="color: #5700f7"> 7 дней – любая точка мира! </p>
```

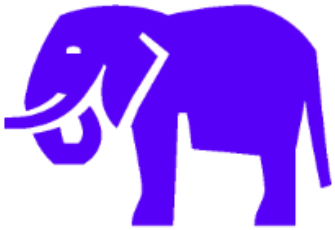
Рисунок 3.1.1.1 – Пример использования набора инструментов Bootstrap

Взаимодействие с серверной частью происходит с помощью модуля Thymeleaf фреймворка Spring, который включает в себя стандартный диалект, включающий в себя работу с атрибутами, синтаксисом, параметризацией, ветвлением, циклами и многим другим (Рисунок 3.1.1.2).

```
<p class="h3 font-weight-normal mt-4 text-center" style="color: #5700f7" th:text="${welcome}"></p>
```

Рисунок 3.1.1.2 – Пример использования модуля Thymeleaf

В случае возникновения какой-либо ошибки, не связанной с полями ввода и формами, а произошедшей, например, при ошибке в ссылке, пользователь получит специальное окно, указывающее на возникновение ошибки (Рисунок 3.1.1.3).



Something went wrong :(

Рисунок 3.1.1.3 – Возникновение ошибки

3.1.2. Межстраничная навигация

Каждая веб-страница имеет навигационное меню, которое оптимизированно как под широкоформатные устройства и планшеты (Рисунок 3.1.2.1), так и под телефоны (Рисунок 3.1.2.2), способное сворачиваться и разворачиваться по нажатии соответствующей кнопки (Рисунок 3.1.2.3).

Рисунок 3.1.2.1 – Навигационное меню

Рисунок 3.1.2.2 – Навигационное меню для телефонов

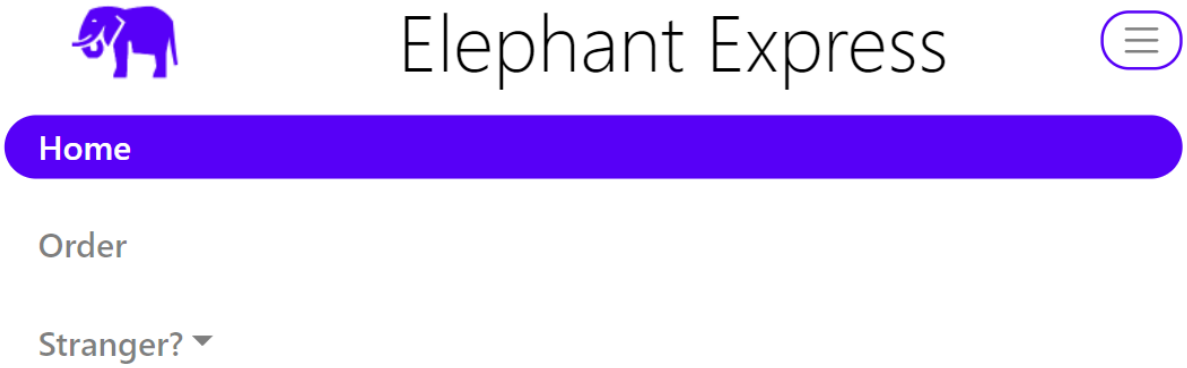


Рисунок 3.1.2.3 – Навигационное меню для телефонов

При реализации межстраничной навигации во время разработки преследовалась цель достижения максимального интуитивного взаимодействия для удобства пользователя. Также межстраничная навигация реализована таким образом, что пользователь всегда может попасть на желаемую веб-страницу, независимо от того, где он находится в текущий момент.

Пользователь всегда понимает, на какой веб-странице приложения он находится, благодаря выделению кнопки перехода на текущую веб-страницу.

При наведении на одну из кнопок переключения на другую веб-страницу интернет ресурса, она выделяется среди остальных, чтобы пользователь точно понимал, что переходит на нужную ему веб-страницу.

В зависимости от ширины экрана устройства навигационная панель адаптивно оптимизируется для удобства пользователя.

3.1.3. Оптимизация веб-страниц для различных браузеров и видов устройств

Несмотря на безусловное удобство, практичность, многозадачность и многофункциональность персональных компьютеров, они с трудом могут в современном мире конкурировать с телефонами, которые благодаря своей компактности и мобильности нивелируют все свои минусы. Сегодня почти каждый человек имеет в своем кармане телефон с доступом в интернет, что делает получение различной информации с множества интернет-ресурсов ещё быстрее и удобнее, если веб-страницы к тому же адаптивно оптимизированы под различные устройства.

Решение задач адаптивной оптимизации веб-страниц и размещаемого контента выполнено для двух основных типов устройств – широкоформатных и узкоформатных, что делает веб-приложение одинаково удобным и практичным как для персональных компьютеров и планшетов (Рисунок 3.1.3.1), так и для телефонов (Рисунок 3.1.3.2).

Основными способами реализации адаптивной оптимизации стали наборы инструментов Bootstrap, которые удобно и быстро решают необходимые задачи.

Тестирование на различных устройствах и в различных браузерах показало, что разбиение под два основных типа устройств является наиболее оптимальным вариантом адаптивной оптимизации.

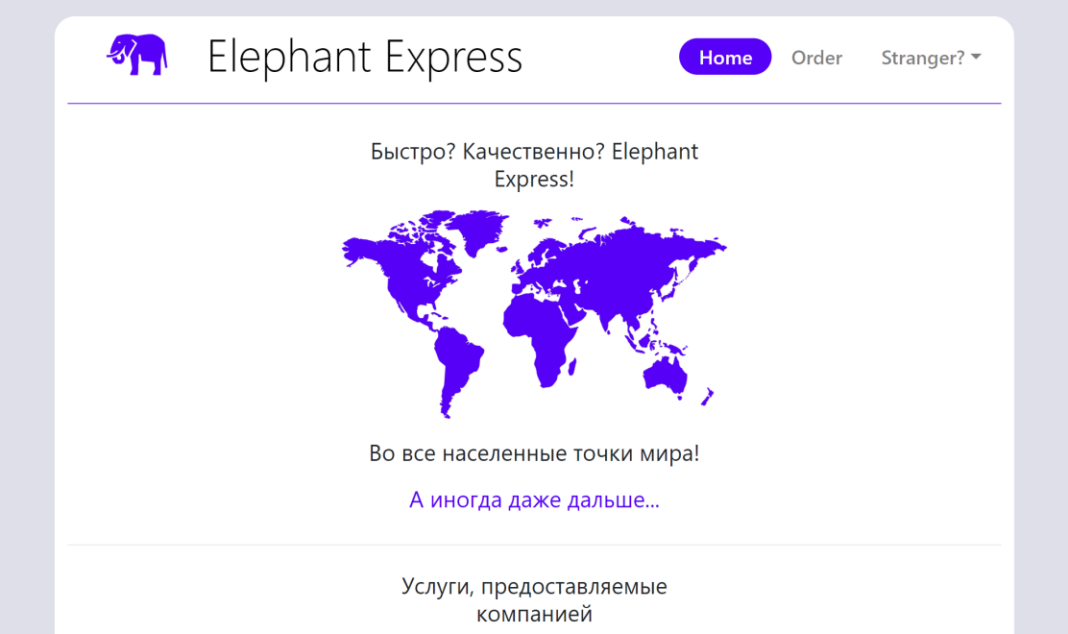


Рисунок 3.1.3.1 – Адаптивная оптимизация под широкоформатные устройства

Рисунок 3.1.3.2 – Адаптивная оптимизация для телефонов

3.2. Структура серверной части веб-приложения

3.2.1. Описание серверной части

Серверная часть веб-приложения расположена в директории «java» в корневом пакете приложения «ru.xpressed.javatemplatescoursework» и разделяется на несколько подпакетов:

- Подпакет «config» используется для хранения конфигурационных файлов приложения.
- Подпакет «controller» используется для хранения различных контроллеров приложения, обрабатывающих запросы.
- Подпакет «entity» используется для хранения сущностей приложения, которые имеют свои поля и методы и сохраняются в базе данных.
- Подпакет «repository» используется для хранения репозиториев приложения, которые работают с базой данных.
- Подпакет «service» используется для хранения сервисных файлов приложения.

Серверная часть веб-приложения реализована с помощью фреймворка Spring языка программирования Java с использованием СУБД PostgreSQL (Рисунок 3.2.1.1).

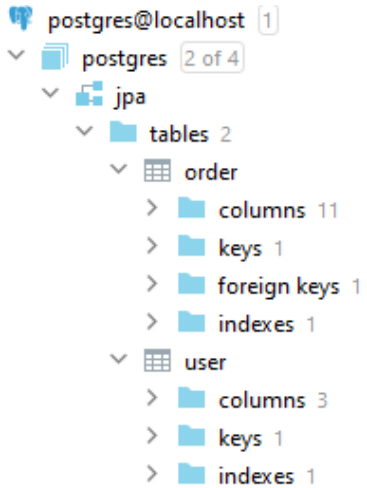


Рисунок 3.2.1.1 – Структура базы данных

Для инициализации Spring приложения использовался встроенный в IntelliJ IDEA функционал (Рисунок 3.2.1.2). Следует указать название приложения, корневую директорию, язык программирования, систему сборки, группы и версию выбранного языка. Тот же результат можно получить в виде архива с официального сайта инициализации Spring.

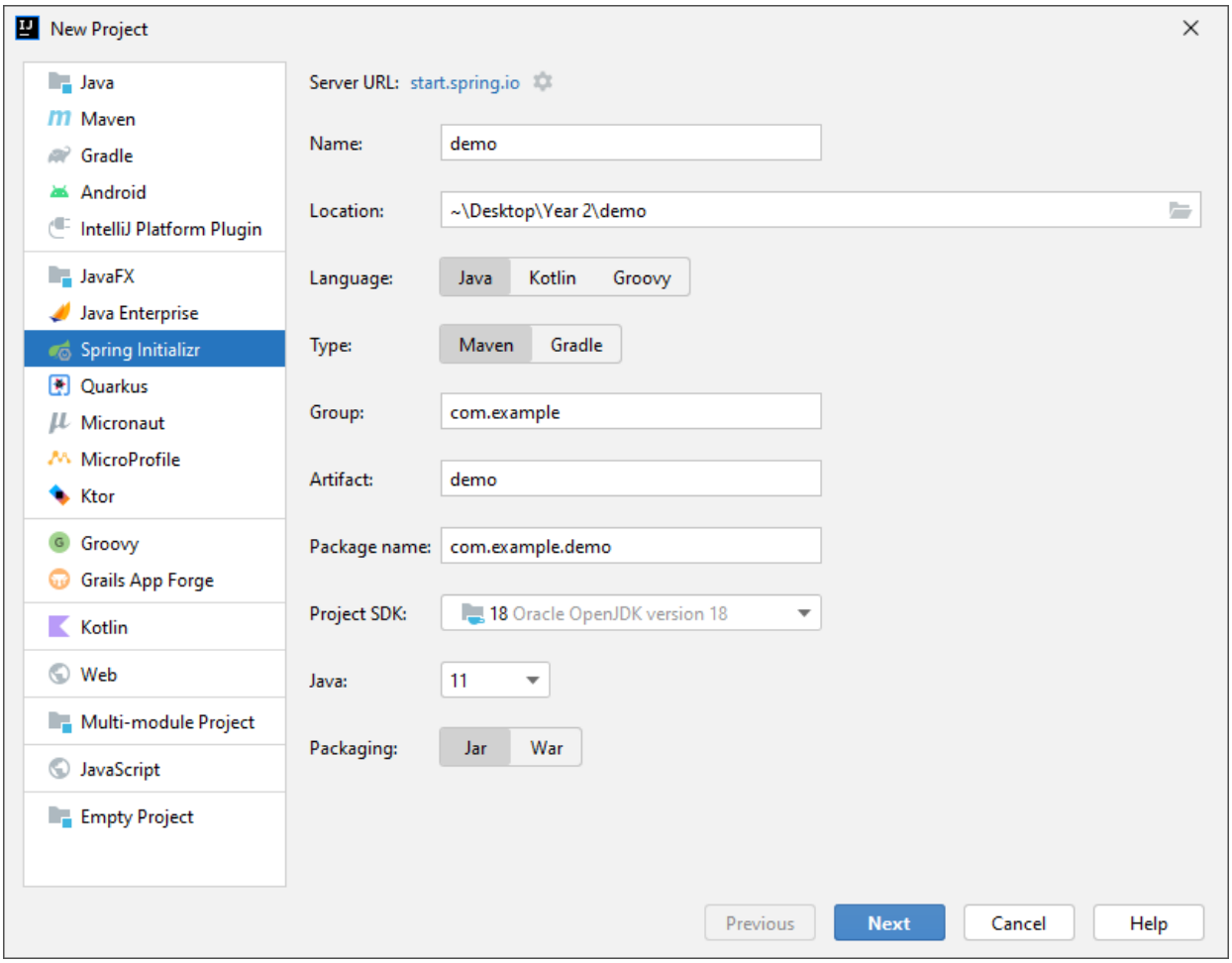


Рисунок 3.2.1.2 – Инициализация Spring приложения

Так же, во время инициализации можно выбрать необходимые модули фреймворка Spring, которые можно будет указать позже в файле настроек выбранной системы сборки (Приложение 1).

Несмотря на то, что файл свойств приложения «application.property» находится в директории «resources», он целиком и полностью относится к

серверной части веб-приложения и необходим для настройки подключения и работы приложения с базой данных (Приложение 1).

3.2.2. Реализация сущностей приложения (Entity)

Реализация всего проекта приложения начинается с создания необходимых сущностей, сразу включая во внимание будущую необходимость сохранения сущностей в базу данных, таблицы которой будут инициализированы автоматически.

- Сущность «Customer» (Рисунок 3.2.2.1) необходима для регистрации и авторизации пользователей, а также сохранения другой персональной информации.

```
package ru.xpressed.javatemplatescoursework.entity;

import lombok.Getter;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Pattern;
import java.util.*;

@Entity
@Table(name = "user")
@Getter
@Setter
public class Customer implements UserDetails {
    @Id
    @NotEmpty(message = "Username can not be empty!")
    @Pattern(regexp = "^[a-zA-Z0-9]{3,15}$", message = "Only numbers and at least 3 and not more than 15 letters!")
    private String username;

    @NotEmpty(message = "Password can not be empty!")
    private String password;

    @Column(updatable = false, insertable = false)
    @NotEmpty(message = "Repeated password can not be empty!")
    private String repeated;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "customer", fetch = FetchType.LAZY)
    public List<Order> orders = new ArrayList<>();

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() { return null; }

    @Override
    public boolean isAccountNonExpired() { return true; }

    @Override
    public boolean isAccountNonLocked() { return true; }

    @Override
    public boolean isCredentialsNonExpired() { return true; }

    @Override
    public boolean isEnabled() { return true; }
}
```

Рисунок 3.2.2.1 – Сущность «Customer»

Как видно из рисунка, сущность «Customer» является имплементацией интерфейса «UserDetails», что пригодиться в будущем при создании безопасности веб-приложения.

Аннотация «@Entity» указывает на то, что данный класс является сущностью.

Аннотация «@Table» необходима для явного указания таблицы базы данных, в которых будут храниться сущности этого класса.

Аннотации «@Getter» и «@Setter», представленные средствами плагина Lombok нужны для автоматического написания шаблонов геттеров и сеттеров для всех полей сущности.

Сущность «Customer» имеет следующие поля:

1. username – имя пользователя, которое также является идентификатором, что указано с помощью аннотации «@Id»;
2. password – пароль пользователя;
3. repeated – повторный пароль пользователя, который необходим во время регистрации для подтверждения пароля, но при этом не сохраняется в базе данных по причине отсутствия необходимости;
4. orders – список заказов пользователя, реализованных сущностью «Order», которые сохраняются в другую таблицу базы данных, но при этом объединены аннотацией «@OneToMany».

Также для каждого поля в отдельности указаны определенные параметры, необходимые для валидации введенной пользователем информации, с помощью аннотаций «@NotEmpty» (Значение не может быть пустым) и «@Pattern» (Значение должно соответствовать определенному паттерну). А также указываются сообщения об ошибке валидации.

Исходя из рисунка, сущность также содержит несколько методов, которые являются перегрузкой методов интерфейса «UserDetails», в которых указываются необходимые далее значения параметров.

- Сущность «Order» (Рисунок 3.2.2.2) необходима для сохранения информации о заказах на доставки, совершенных определенным пользователем.

```
package ru.xpressed.javatemplatescoursework.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

@Entity
@Getter
@Setter
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @NotEmpty(message = "Loading point can not be empty!")
    private String load;

    @NotEmpty(message = "Destination point can not be empty!")
    private String destination;

    @NotEmpty(message = "Contact information can not be empty!")
    private String contact;

    private String info;

    private SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd.MM.y HH:mm");

    private String departure;

    private String arrival;

    private Date arrivalDate;

    private int btn;

    public void setDeparture(Date departure) {
        this.departure = dateFormat.format(departure);

        Calendar cal = Calendar.getInstance();
        cal.setTime(departure);
        cal.add(Calendar.MINUTE,  amount: 2);
        this.arrivalDate = cal.getTime();
        this.arrival = dateFormat.format(cal.getTime());
    }

    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.DETACH)
    private Customer customer;
}
```

Рисунок 3.2.2.2 – Сущность «Order»

Как видно из рисунка сущность «Order» имеет схожесть в использовании аннотаций с сущностью «Customer», в связи с чем описание части аннотаций можно пропустить.

Сущность «Customer» имеет следующие поля:

1. id – идентификатор заказа;
2. load – адрес, указанный пользователем для отправки заказа;
3. destination – адрес, указанный пользователем для доставки заказа;
4. contact – некоторая контактная информация пользователя;
5. info – любая дополнительная информация, не являющаяся обязательной;
6. dateFormat – некоторый формат представления даты и времени;
7. departure – дата и время отправления заказа;
8. arrival – расчетные дата и время прибытия заказа;
9. customer – пользователь, за которым закреплен заказ, объединенный аннотацией «@ManyToOne».

Также у сущности имеется некоторый метод «setDeparture», который переводит из даты в строку по указанному ранее формату значения даты и времени оправки и прибытия.

В этом случае идентификатором сущности является некоторое числовое значение, которое генерируется автоматически с помощью аннотации «@GeneratedValue».

3.2.3. Реализация репозиториев приложения (Repository)

После реализации сущностей, необходимо выполнить реализацию репозиториев, необходимых для работы серверной части приложения с базой данных.

Все репозитории являются наследниками класса «JpaRepository», предоставляемого модулем фреймворка Spring, а именно Spring Data JPA,

который осуществляет работу с базой данных, включая сохранение, удаление и поиск сущностей в таблицах.

Репозиторий «CustomerRepository» (Рисунок 3.2.3.1) необходим для работы с таблицей, в которой хранятся сущности «Customer».

```
package ru.xpressed.javatemplatescoursework.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.xpressed.javatemplatescoursework.entity.Customer;

@Repository
public interface CustomerRepository extends JpaRepository<Customer, Integer> {
    Customer findByUsername(String username);
}
```

Рисунок 3.2.3.1 – Репозиторий «CustomerRepository»

Как видно из рисунка, все репозитории являются интерфейсами и помечаются специальной аннотацией «@Repository».

Также данный репозиторий имеет лишь один дополнительный метод «findByUsername», который имеет определенной название и необходим для поиска сущности «Customer» по его идентификатору («username»).

Репозиторий «OrderRepository» (Рисунок 3.2.3.2) необходим для работы с таблицей, в которой хранятся сущности «Order».

```
package ru.xpressed.javatemplatescoursework.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.xpressed.javatemplatescoursework.entity.Order;

@Repository
public interface OrderRepository extends JpaRepository<Order, Integer> {
}
```

Рисунок 3.2.3.2 – Репозиторий «OrderRepository»

Как видно из рисунка, репозиторий «OrderRepository» не имеет дополнительных методов.

3.2.4. Реализация контроллеров приложения (Controller)

После реализации репозиториев необходимо реализовать различные контроллеры, которые будут обрабатывать серверные запросы.

Работа всех контроллеров выполняется по аналогичному сценарию, поэтому имеет смысл описать лишь один семи существующих контроллеров в проекте приложения.

Контроллер «RegistrationController» (Рисунок 3.2.4.1) необходим для обработки запросов по валидации и регистрации новых пользователей в базе данных.

```java
package ru.xpressed.javatemplatescoursework.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import ru.xpressed.javatemplatescoursework.config.SecurityConfig;
import ru.xpressed.javatemplatescoursework.entity.Customer;
import ru.xpressed.javatemplatescoursework.repository.CustomerRepository;

import javax.validation.Valid;

@Controller
public class RegistrationController {
    @Autowired
    CustomerRepository customerRepository;
    @Autowired
    SecurityConfig securityConfig;

    @GetMapping("/registration")
    public String showRegistrationForm(Customer customer, Model model) {
        model.addAttribute( attributeName: "customer", new Customer());
        return "registration";
    }

    @PostMapping("/registration")
    public String completeRegistration(@Valid Customer customer, BindingResult bindingResult, Model model) {
        if (bindingResult.hasErrors()) {
            model.addAttribute( attributeName: "customer", customer);
            return "registration";
        }

        if (!customer.getPassword().equals(customer.getRepeated())) {
            model.addAttribute( attributeName: "reperr",  attributeValue: "Passwords don't match!");
            return "registration";
        }

        if (customerRepository.findByUsername(customer.getUsername()) == null) {
            customer.setPassword(securityConfig.encoder().encode(customer.getPassword()));
            customerRepository.save(customer);
            model.addAttribute( attributeName: "message",  attributeValue: "Registration Completed");
            model.addAttribute( attributeName: "onload",  attributeValue: "redirectTimer()");
        } else {
            bindingResult.rejectValue( field: "username",  errorCode: "customer.username",
                    defaultMessage: "This username is already taken!");
        }
        return "registration";
    }
}
```

Рисунок 3.2.4.1 – Контроллер «RegistrationController»

Как видно из рисунка, все контроллеры помечаются специальной аннотацией «@Controller», явно указывающей на то, что данный класс является контроллером.

Репозитории и другие классы, которые участвуют в работе контроллера необходимо «связать» с данным контроллером с помощью аннотации «@Autowired».

Далее необходимо указать контроллеру, какие запросы он должен выполнять с помощью аннотации «@GetMapping» и указания URL. Далее необходимо реализовать функцию, в теле которой и будет происходить обработка запроса и которая будет возвращать HTML страницу в стандартном или измененном с помощью модуля Thymeleaf фреймворка Spring виде.

Иногда в контроллерах используется аннотация «@PostMapping», которая указывает, что функцию обработки необходимо вызвать после «@GetMapping» и определенных действий. В данном случае такая функция используется для валидации данных, введенных пользователем и сохранения нового пользователя в базе данных или уведомления об ошибке.

3.2.5. Реализация сервисов приложения (Service)

Сервисы необходимы для осуществления бизнес-логики серверной части. В данном проекте используется только один сервис «CustomerService» (Рисунок 3.2.5.1), необходим для загрузки пользователей из базы данных по их идентификатору («username»).

```
package ru.xpressed.javatemplatescoursework.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import ru.xpressed.javatemplatescoursework.entity.Customer;
import ru.xpressed.javatemplatescoursework.repository.CustomerRepository;

@Service
public class CustomerService implements UserDetailsService {
    @Autowired
    CustomerRepository customerRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Customer customer = customerRepository.findByUsername(username);

        if (customer == null)
            throw new UsernameNotFoundException("User not found!");

        return customer;
    }
}
```

Рисунок 3.2.5.1 – Сервис «CustomerService»

Как видно из рисунка, все сервисы помечаются специальной аннотацией «@Service».

3.2.6. Обработка серверных запросов

Обработка серверных запросов веб-приложения происходит по определенному сценарию, в котором участвуют до всех подпакетов приложения (Рисунок 3.2.6.1).

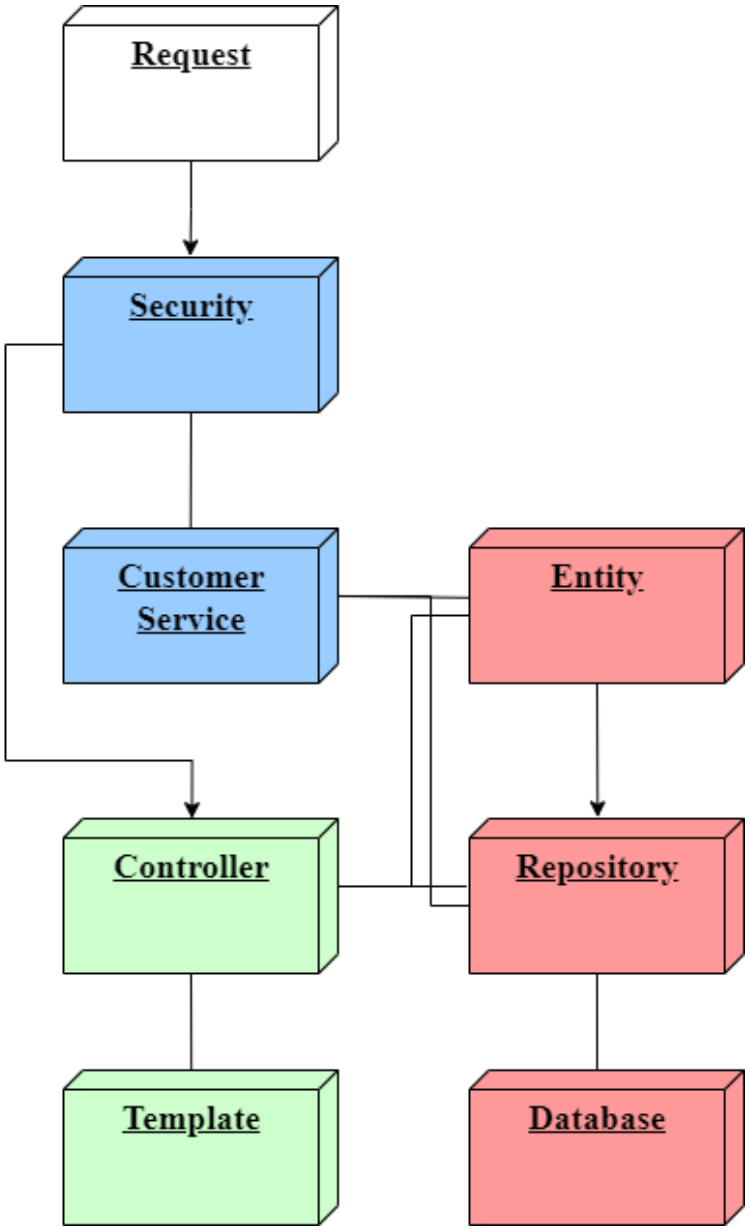


Рисунок 3.2.6.1 – Визуализация обработки запросов

4. ТЕСТИРОВАНИЕ

Тестирование веб-приложения заключалось в успешном выполнении следующих задач, которые в той или иной степени могут повлечь за собой сбои различных категорий в работе приложения или базы данных, путем правильной реакции со стороны клиентской или серверной логики на определенные действия со стороны пользователя:

- Проверка адаптивности веб-страниц для различных браузеров и типов устройств.
- Проверка на ввод данных для различных полей ввода и форм, значение которых должно находится в рамках определенных шаблонов.
- Проверка работы веб-приложения в LAN и WAN сетях.
- Проверка реакции веб-приложения на попытки обхода безопасности.
- Проверка веб-приложения на обработку других ошибок.

Проверка адаптивности всех веб-страниц выполнялась на большом количестве устройств и браузеров, включая не распространенные варианты форматов, разрешений и типов систем.

Проверка на ввод данных выполнялась для всех полей ввода и форм, при этом разными людьми и способами (Рисунок 4.1.1, рисунок 4.1.2, рисунок 4.1.3).

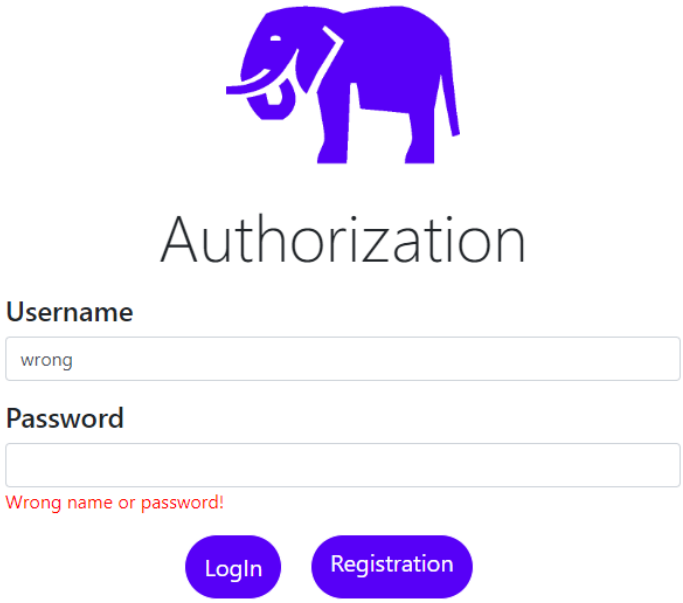


Рисунок 4.1.1 – Тестирование авторизации

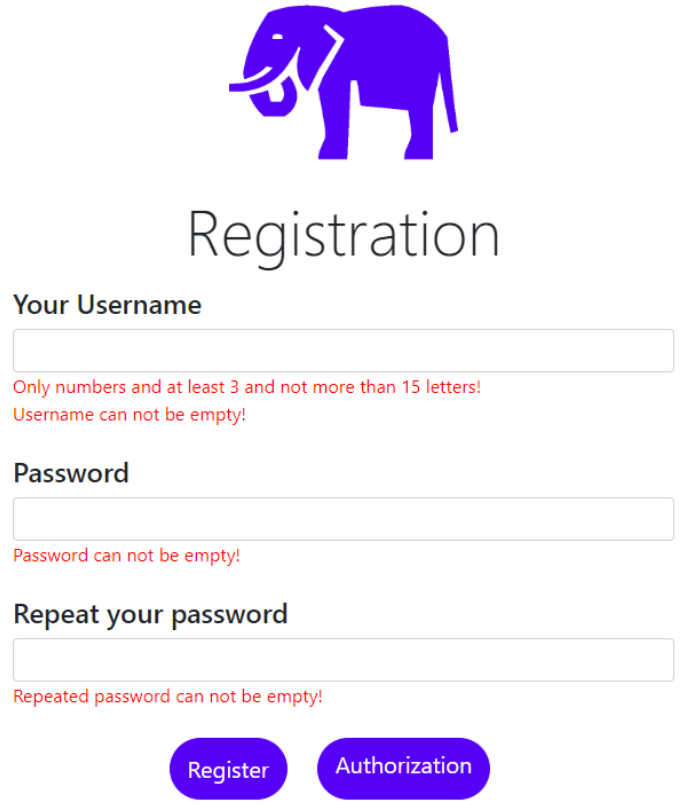


Рисунок 4.1.2 – Тестирование регистрации

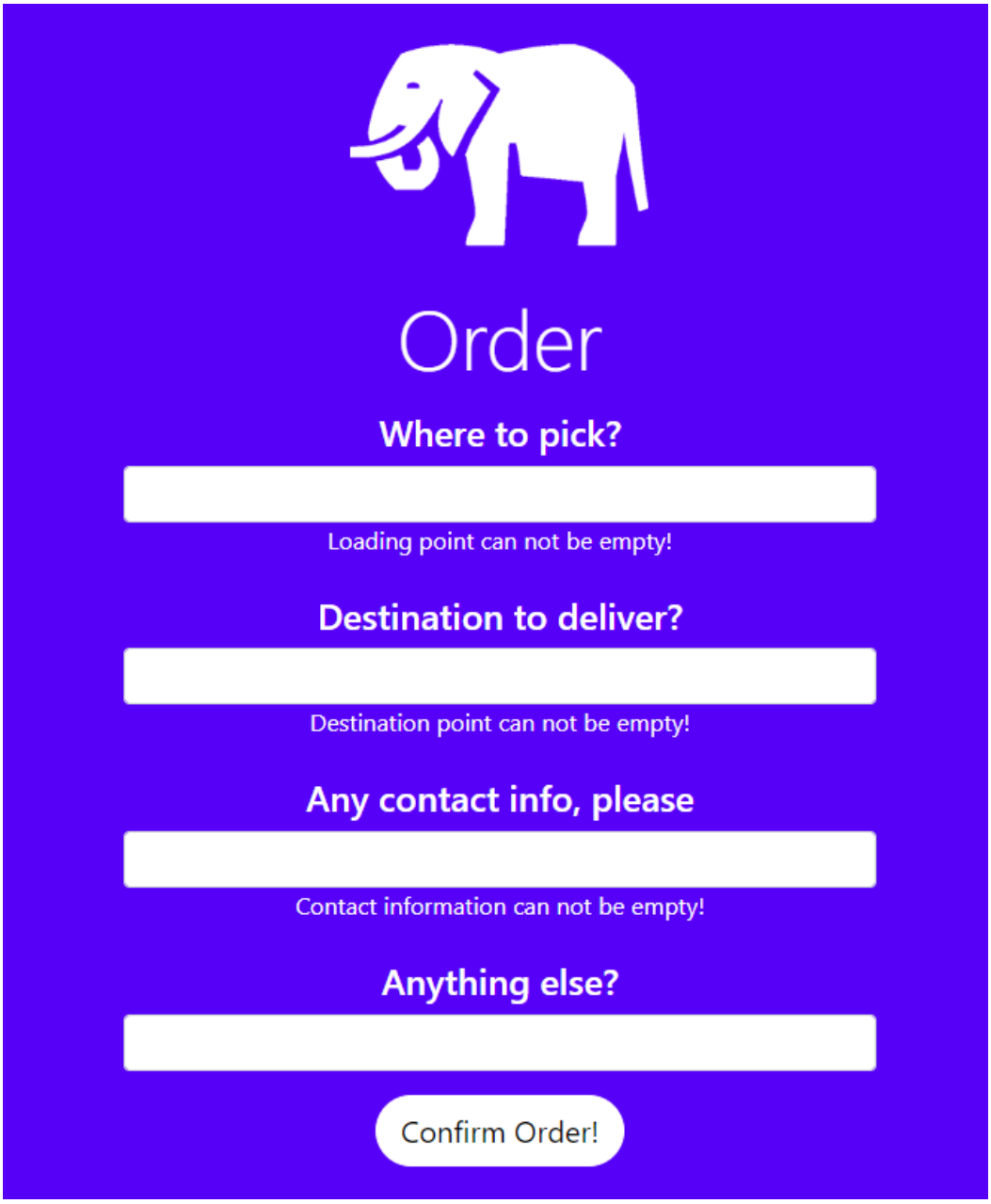


Рисунок 4.1.3 – Тестирование заказа

Проверка работы в LAN сети проводилась сразу на этапе разработки приложения, поскольку эта сеть использовалась для тестирования новых компонентов приложения. Проверка работы в WAN сети проводилась после завершения этапа реализации и производилась разными людьми из разных точек. Подключение из WAN производилось путем проброса порта.

Проверка реакции веб-приложения на различные попытки обхода безопасности была выполнена путем попыток подбора пароля, подмены аккаунта и подмены адреса.

Проверка веб-приложения на обработку других ошибок производилась путем доступа к несуществующим веб-страницам.

В ходе тестирования было внесено некоторое количество необходимых изменений, и, в результате, все проверки были пройдены без каких-либо оставшихся замечаний.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы были выполнены все поставленные задачи и была достигнута основная цель по разработке, тестированию и отладке практичного веб-приложения на тему «Фирма грузоперевозок» с применением различных модулей фреймворка Spring, организацией межстраничной навигации, оптимизацией веб-страниц и размещаемого контента для различных браузеров и видов устройств согласно поставленным требованиям.

Веб-приложения доступно по персональному динамическому домену. Проект веб-приложения доступен на публичном репозитории.

Ссылка на веб-приложение – http://johnswhome.ddns.net:1337/.

Ссылка на репозиторий с проектом веб-приложения на GitHub – https://github.com/xPressed/JavaTemplates-CourseWork.

В процессе работы над курсовой работой были приобретены следующие компетенции:

- ПК-1 – способен выполнять разработку и интеграцию программных модулей и компонент, верификацию выпусков программных продуктов информационных систем;
- ПК-1.1 – Знать: методы и средства сборки модулей и компонент программного обеспечения при создании информационных систем;
- ПК-1.12 – Уметь: применять методы и средства создания программных интерфейсов информационных систем;
- ПК-1.14 – Владеть: разработкой процедур сборки модулей и компонент программного обеспечения при внедрении информационных систем

Отчет сформирован согласно инструкции по организации и проведению курсового проектирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Spring [Электронный ресурс]. — URL: https://spring.io/ (Дата последнего обращения: 03.05.2021

2. Baeldung [Электронный ресурс]. — URL: https://www.baeldung.com/ (Дата последнего обращения: 04.05.2022)

3. Stack Overflow [Электронный ресурс]. — URL: https://stackoverflow.com/ (Дата последнего обращения 04.05.2022)

4. Bootstrap [Электронный ресурс]. — URL: https://getbootstrap.com/ (Дата последнего обращения: 05.05.2022)

5. ICONS8 [Электронный ресурс]. — URL: https://icons8.com/ (Дата последнего обращения: 01.05.2022)

ПРИЛОЖЕНИЕ 1

Листинг 1 – Файл настройки системы сборки Gradle

```
plugins {
    id 'org.springframework.boot' version '2.6.7'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'ru.xpressed'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = "18.0.1"

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity5'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    runtimeOnly 'org.postgresql:postgresql'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

Листинг 2 – Файл свойств веб-приложения

```
# Connection
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=123

# JPA
spring.jpa.properties.hibernate.default_schema=jpa
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.database=POSTGRESQL
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy
spring.jpa.properties.hibernate.id.new_generator_mappings=true
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```