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 «Цветочный магазин»

Студент: Шумахер Марк Евгеньевич
Группа: ИКБО-20-22
Работа представлена к защите_____(дата)_____/Шумахер М.Е./
(подпись и ф.и.о. студента)

Руководитель: старший преподаватель Рачков Андрей Владимирович

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

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

М. РТУ МИРЭА. 2024

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
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Шумахер Марк Евгеньевич
Группа: ИКБО-20-22
Срок представления к защите: 13.05.2024 г.
Руководитель: старший преподаватель Рачков Андрей Владимирович

Тема: Приложение «Цветочный магазин»

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 в соответствии с ГОСТ 7.32-2017. 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/, «28» февраля 2024 г.
Задание на КР выдал: _____/ А.В. Рачков /, «28» февраля 2024 г.
Задание на КР получил: _____/ М.Е. Шумахер /, «28» февраля 2024 г.

РЕФЕРАТ

Отчет 25 с., 22 рис., 11 источн.

ИНТЕРНЕТ, РАЗРАБОТКА СЕРВЕНОЙ ЧАСТИ ПРИЛОЖЕНИЯ, ВЕБ-ПРИЛОЖЕНИЕ, SPRING, SPRING BOOT, DOCKER, POSTGRESQL, GRADLE, СОВРЕМЕННЫЕ ТЕХНОЛОГИИ, ЦВЕТЫ

Объект исследования – веб-приложение «Цветочный магазин».

Предмет исследования – разработка серверной части веб-приложения «Цветочный магазин».

В ходе работы был проведен анализ предметной области и обзор веб-приложений с аналогичной тематикой.

Рассмотрен процесс создания веб-приложения, используемый программный инструментарий и среда разработки.

Результатом работы является веб-приложение «Цветочный магазин», учитывая постоянную потребность цветочных композициях, предполагается, что в перспективе число пользователей веб-приложения будет расти.

REPORT

Report 25 p., 22 fig., 11 sources.

INTERNET, DEVELOPMENT OF THE SERVER PART OF THE APPLICATION, WEB APPLICATION, SPRING, SPRING BOOT, DOCKER, POSTGRESQL, GRADLE, MODERN TECHNOLOGIES, FLOWERS

The object of the study is the «Flower Shop» web application.

The subject of the study is the development of the server part of the Flower Shop web application.

In the course of the work, a domain analysis and a review of web applications with similar topics were carried out.

The process of creating a web application, the software tools used and the development environment are considered.

The result of the work is the web application «Flower Shop», given the constant need for flower arrangements, it is assumed that in the future the number of users of the web application will grow.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете применяют следующие термины с соответствующими определениями:

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта

Хостинг – услуга по предоставлению ресурсов для размещения информации на сервере, постоянно имеющем доступ к сети

Dockerfile – файл для предварительной работы, набор инструкций, который нужен для записи образа

Render – бесплатный хостинг для размещения приложений

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете применяются следующие сокращения и обозначения:

HTML – HyperText Markup Language, язык гипертекстовой разметки

JPA – Java Persistence API

JS – JavaScript, язык программирования

UML – Unified Modeling Language

ВВЕДЕНИЕ

Целью курсовой работы является процесс разработки серверной части веб-приложения «Цветочный магазин», а также анализ предметной области с целью определения основных требований к приложению. Основная задача заключается в разработке собственной архитектуры и функционала приложения, используя язык программирования Java [1], фреймворк Spring [2] и базу данных PostgreSQL [3]. Для тестирования функционала приложения будет использоваться инструмент Postman [4], а для контейнеризации Docker [5]. Для достижения поставленной цели были сформулированы следующие задачи:

– провести анализ предметной области, связанной с разрабатываемым приложением,

– выбрать инструменты для ведения разработки,

– разработать приложение с использованием выбранных средств разработки,

– протестировать разработанное приложение.

Для выполнения этих задач применяются методы анализа и сравнительного исследования. Информационной базой для этой работы являются знания, полученные в ходе практических занятий курса «Шаблоны программных платформ на языке Java», а также данные из интернет-ресурсов.

В разделе описания логической структуры приведены методы анализа, которые были применены в ходе работы, и структура базы данных приложения.

В разделе, посвященном обоснованию выбора технологий, представлена информация о использованных программных инструментах.

Раздел, посвященный разработке приложения, содержит информацию о созданном приложении, его тестировании и общей структуре.

В разделе заключения подводятся итоги по выполненной разработке и приводятся ссылки на исходный код и хостинг веб-приложения.

1 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

1.1 Анализ предметной области

Анализ предметной области проводился среди веб-приложений на тематику «Цветочный магазин». Были выбраны три веб-приложения: Flowwow [6], простоцветы [7] и Мега Цветы [8], представленные на рисунках 1-3.

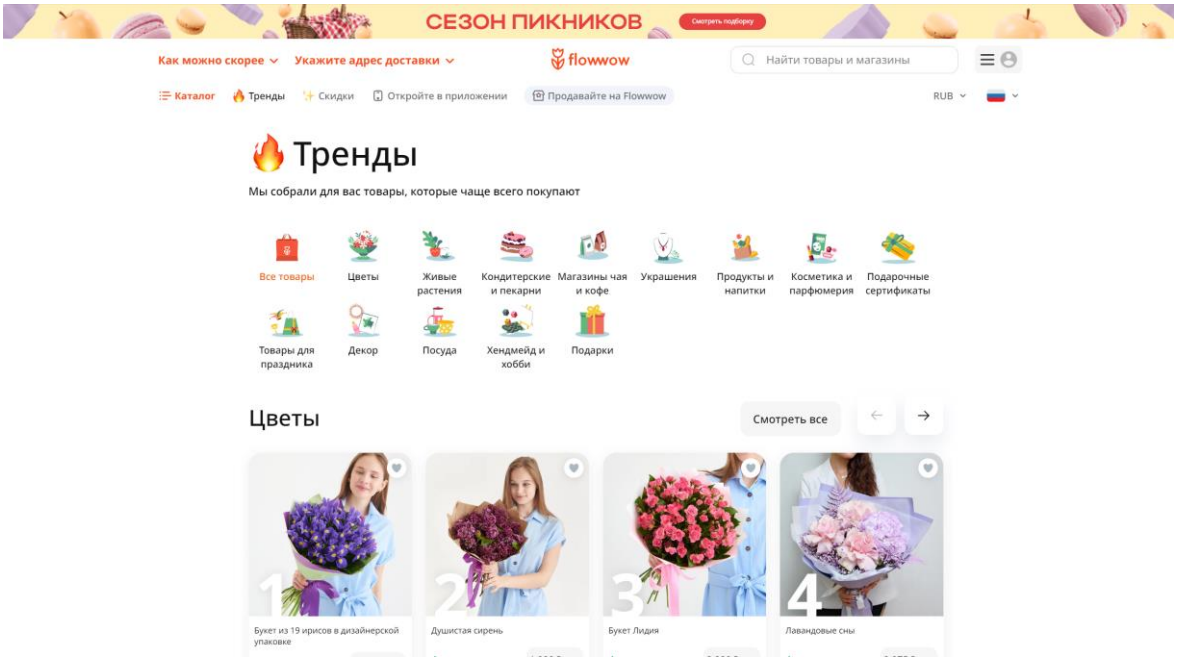

Рисунок 1 – Клиентская часть сайта flowwow

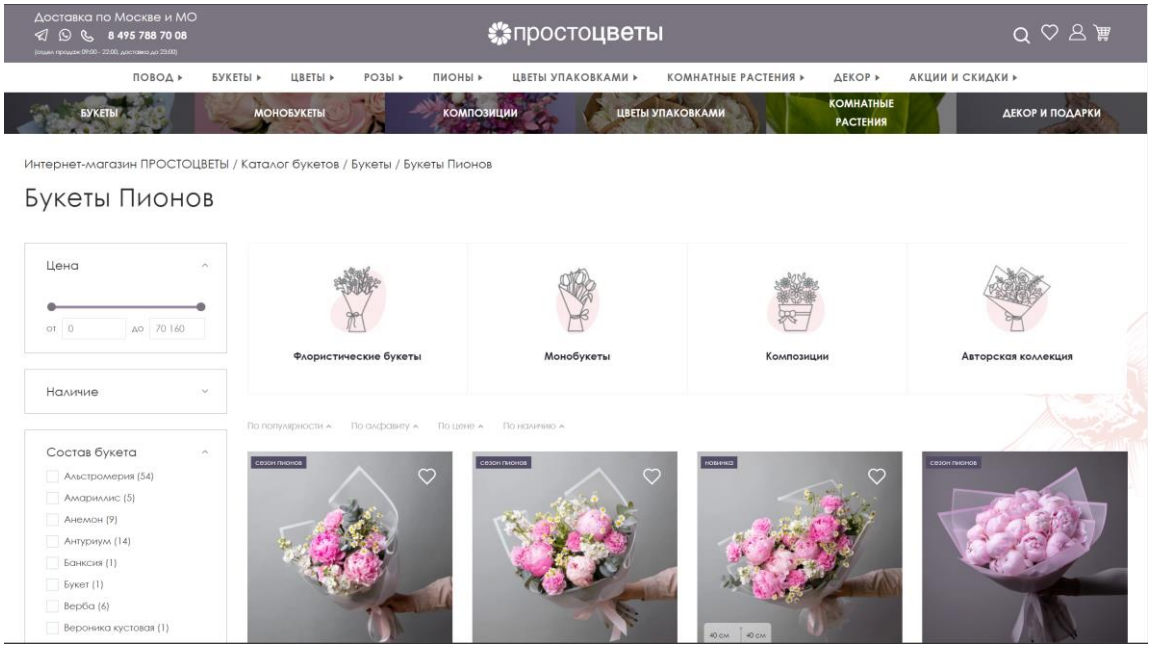

Рисунок 2 – Клиентская часть сайта простоцветы

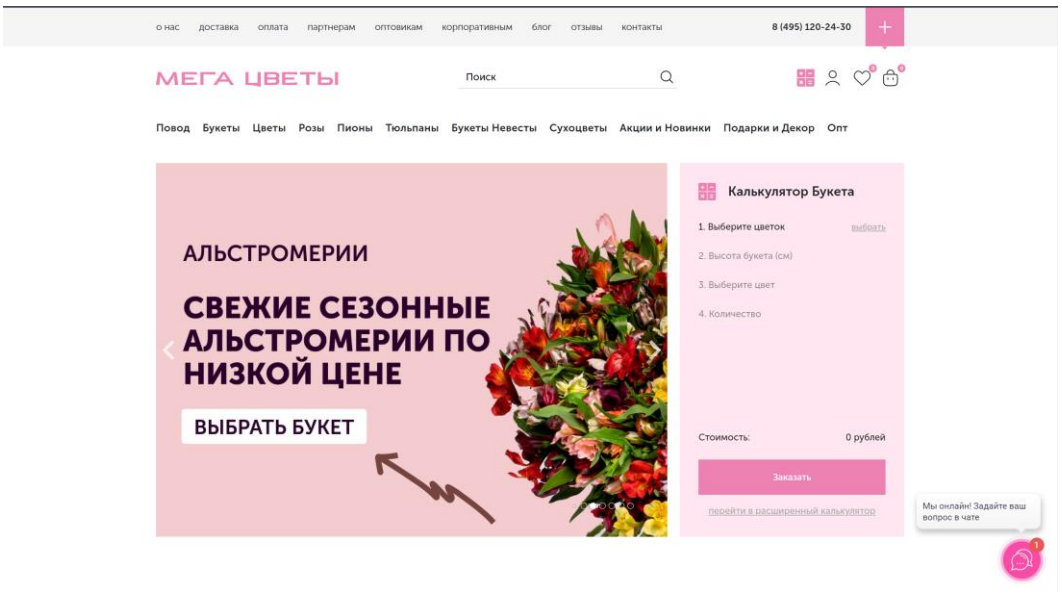

Рисунок 3 – Клиентская часть сайта Мега Цветы

Все упомянутые веб-приложения включают функционал регистрации и входа в аккаунт, возможность просмотра товаров, добавление их в корзину, оформление заказа на основе содержимого корзины, а также удобный просмотр истории заказов пользователя.

Из анализа данных веб-приложений следует выделить ключевые возможности, которыми должен обладать разрабатываемый веб-ресурс данной тематики:

– регистрация нового аккаунта,

– вход в существующий аккаунт,

– выход из аккаунта,

– просмотр товаров,

– добавление товаров в корзину,

– оформление заказа,

– просмотр заказов внутри личного кабинета.

1.2 Структура базы данных

Для хранения пользователей, с хешированным паролем, товаров и заказов веб-приложения, используется база данных в определенной структурой, описанной с помощью UML-диаграмм на рисунке 4.

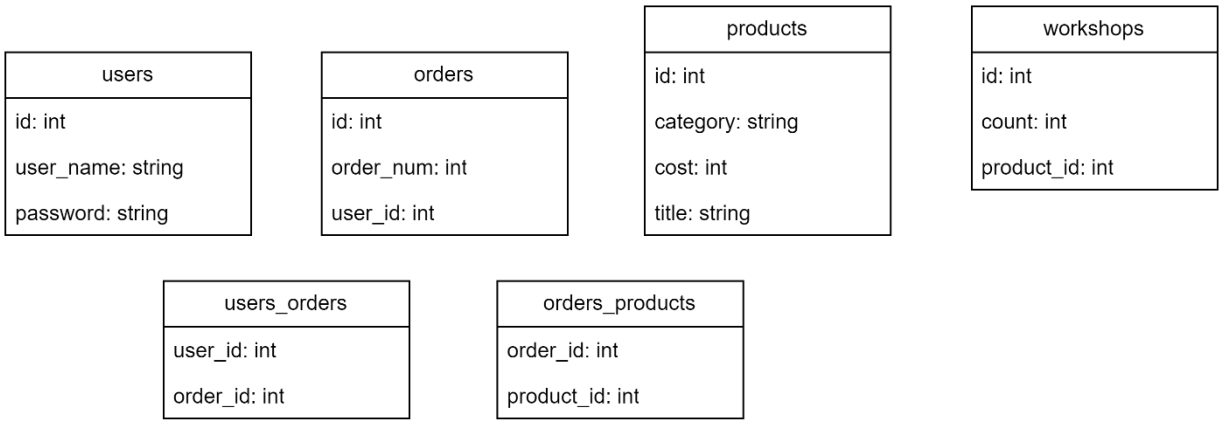

Рисунок 4 – Структура базы данных приложения

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

2.1 Используемое прикладное программное обеспечение

Для разработки приложения использовалась интегрированная среда разработки программного обеспечения IntelliJ IDEA, разработанная компанией JetBrains.

2.2 Используемые технологии

При разработке приложения был использован язык программирования Java, система автоматической сборки Gradle [8], фреймворк Spring с его дополнительными инструментами, такими как Spring Data JPA для работы с базой данных (который является надстройкой над Hibernate) и Spring Security для обеспечения функционала регистрации и авторизации. Для создания динамических HTML страниц был использован шаблонизатор Thymeleaf [9]. В качестве базы данных применялась PostgreSQL. Для тестирования использовался Postman, а для развертывания – Docker. Приложение размещено на хостинге Render [10], включая базу данных.

Java - язык программирования, который обеспечивает возможность разработки приложений для различных устройств и платформ, включая персональные компьютеры, мобильные устройства и серверы, благодаря своей многоцелевой природе.

Gradle – это мощный инструмент автоматизации сборки с открытым исходным кодом, который обеспечивает быструю и надежную работу. Его гибкий декларативный язык сборки позволяет автоматизировать разнообразные сценарии создания программного обеспечения, что делает его привлекательным выбором для разработчиков.

Spring представляет собой полноценную модель разработки и настройки для современных корпоративных приложений, созданных на базе языка программирования Java, что позволяет развертывать их на различных платформах. Основной фокус Spring – это инфраструктурная поддержка на уровне приложений, что дает возможность разработчикам избежать привязки к конкретным средам развертывания.

Spring Data JPA, в составе Spring Data, упрощает создание репозиториев на основе Java Persistence API (JPA), что существенно упрощает разработку приложений на Spring, использующих доступ к данным.

Spring Security предоставляет эффективные и гибко настраиваемые инструменты для аутентификации и управления доступом в Java приложениях. Его ключевым преимуществом является легкость расширения, что позволяет удовлетворять индивидуальные требования проекта.

Thymeleaf – это инновационный серверный механизм шаблонов на языке программирования Java, который разработан для удобного создания качественных макетов веб-страниц и автономных приложений. Его особенности включают широкий набор функций и гибкую настройку, делая его привлекательным выбором для разработчиков.

PostgreSQL представляет собой мощную и бесплатную систему управления базами данных с открытым исходным кодом. Эта система предоставляет разработчикам и администраторам возможность создавать, хранить и эффективно обрабатывать данные, используя стандартный язык запросов – SQL (Structured Query Language). Благодаря своей гибкости и расширяемости, PostgreSQL широко применяется в различных проектах, включая веб-приложения, корпоративные системы и аналитические платформы.

Postman - инструмент для тестирования API, который предоставляет широкий функционал, включая отправку запросов, создание документации, запуск автотестов и управление версиями.

Docker – это платформа для упаковки приложений в контейнеры со всеми необходимыми зависимостями для их запуска в различных средах.

Render – это комплексная облачная платформа, предназначенная для разработки, развертывания и управления приложениями и веб-сайтами. Она обеспечивает автоматическое развертывание приложений из репозитория Git.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Структура приложения

Рабочая директория приложения содержит в себе несколько уровней. Серверная часть находиться в директории java/com/flowershop, а клиентская часть, код для страниц, стили, скрипты и изображения, в директории resourses. Подробнее рабочая директория изображена на рисунке 5.

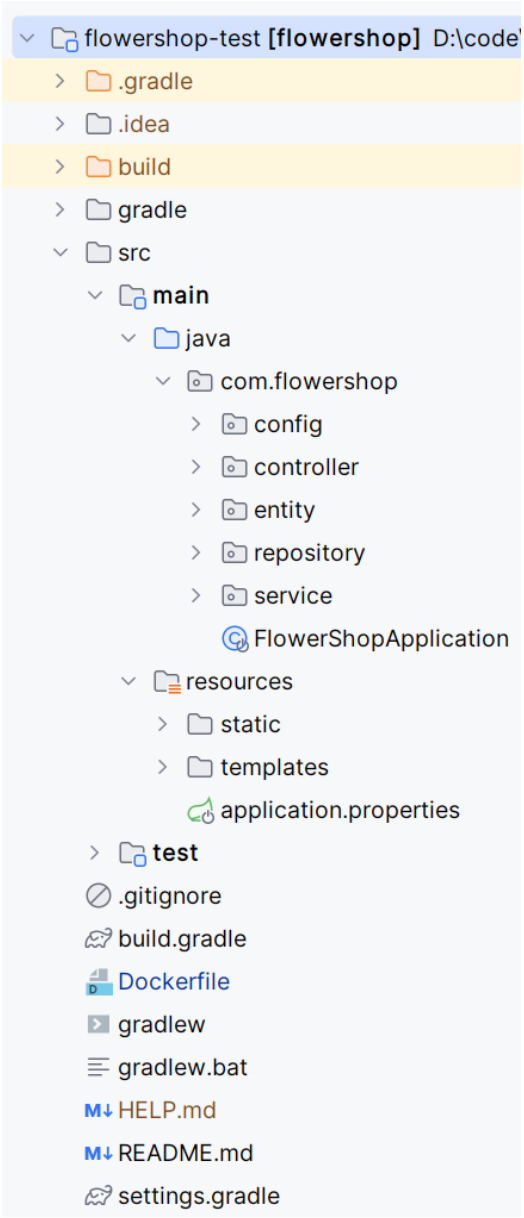

Рисунок 5 – Рабочая директория приложения

3.2 Подключение к базе данных

Для осуществления подключения к базе данных приложения требуется внести соответствующие настройки в файл application.properties, как показано

на рисунке 6. Эти настройки включают строку подключения, а также учетные данные пользователя и пароль для доступа к конкретной базе данных. В контексте данного приложения используется база данных, размещенная на хостинге, поэтому все необходимые данные для подключения предоставляются самим хостингом.

```
spring.application.name=FlowerShop
server.port=8081
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://dpg-corvfs21hbls73fc7ac0-a.frankfurt-postgres.render.com/flowershop_7oyy
spring.datasource.username=flowershop_7oyy_user
spring.datasource.password=VfYl4Y0wLNm53lIJE7PVUX0zjhEOloV0
spring.mvc.static-path-pattern=/resources/**
```

Рисунок 6 – Файл application.properties

3.3 Разработка серверной части приложения

Процессы авторизации и регистрации в приложении реализованы с использованием фреймворка Spring Security. Для обеспечения функциональности этих процессов необходимо настроить соответствующий файл конфигурации, представленный на рисунке 7.

```
Mark
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception
{
    return http.csrf(AbstractHttpConfigurer::disable).cors(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests((auth) ->
                    auth.requestMatchers("/login", "/register", "/", "/resources/**").permitAll()
                    .anyRequest().authenticated())
            .formLogin(form -> form
                    .loginPage("/login")
                    .defaultSuccessUrl("/catalog")
                    .permitAll())
            .logout(logout -> logout
                    .logoutUrl("/logout")
                    .logoutSuccessUrl("/")
                    .invalidateHttpSession(true)
                    .deleteCookies( ...cookieNamesToClear: "JSESSIONID", "cartItems"))
            .build();
}
```

Рисунок 7 – Метод настройки авторизации и регистрации

Для обработки конечных точек авторизации/регистрации написан контроллер EntryController, показанный на рисунке 8.

```
Mark
@GetMapping("/register")
public String regGet(Model model)
{
    model.addAttribute( attributeName: "user", new UserDTO());
    return "register";
}

Mark
@PostMapping("/register")
public String regPost(@ModelAttribute("user") UserDTO newUser, HttpServletRequest request) throws ServletException
{
    if (!newUser.getPassword().equals(newUser.getConfirmPassword()))
    {
        return "redirect:/register?error";
    }

    if (entryService.addUser(newUser))
    {
        request.login(newUser.getName(), newUser.getPassword());
        return "redirect:/catalog";
    }

    return "redirect:/register?exists";
}

Mark
@GetMapping("/login")
public String loginPage()
{
    return "login";
}
```

Рисунок 8 – Контроллер для входа и регистрации пользователя

После входа/регистрации, пользователь перенаправляется на страницу с товарами, находящуюся по адресу /catalog. На рисунке 9 показана обработка данной конечной точки.

```
Mark
@GetMapping("/catalog")
public String catalogGet(Model model)
{
    var workshops = workshopService.getWorkshops();
    model.addAttribute( attributeName: "workshops", workshops);
    return "catalog";
}
```

Рисунок 9 – Обработка вывода товаров

При выборе товаров они попадают в корзину, которая обрабатывается функциями на JS, рисунок 10.

```
4 usages   Mark *
function addToCart(button) : void  {
    const cartItemsElement : HTMLElement  = document.getElementById( elementId: 'cart-items');
    const productName : any  = button.closest('.product').querySelector('.productName').textContent;
    const productCost : any  = button.closest('.product').querySelector('.productCost').textContent;
    const existingCartItem : Element  = findCartItemByName(productName);

    if (existingCartItem)
    {
        increaseQuantity(existingCartItem);
    }
    else
    {
        const cartItemElement : HTMLLIElement  = document.createElement( tagName: 'li');
        cartItemElement.textContent = productName;
        cartItemElement.classList.add('cart-item');
        const quantityControls : HTMLDivElement  = document.createElement( tagName: 'div');
        quantityControls.classList.add('quantity-controls');
        const decreaseButton : HTMLButtonElement  = document.createElement( tagName: 'button');
        decreaseButton.textContent = '-';
        decreaseButton.onclick = function() : void  { decreaseQuantity(cartItemElement); };
        const quantitySpan : HTMLSpanElement  = document.createElement( tagName: 'span');
        quantitySpan.textContent = '1';
        const increaseButton : HTMLButtonElement  = document.createElement( tagName: 'button');
        increaseButton.textContent = '+';
        increaseButton.onclick = function() : void  { increaseQuantity(cartItemElement); };
        quantityControls.appendChild(decreaseButton);
        quantityControls.appendChild(quantitySpan);
        quantityControls.appendChild(increaseButton);
        cartItemElement.appendChild(quantityControls);
        cartItemsElement.appendChild(cartItemElement);
    }
    saveCartToCookies();
}
```

Рисунок 10 – Добавление в корзину товаров

После заполнения корзины товарами, пользователей может оформить заказ, доставая список из хранилища и передавая его на сервер. Контроллер для обработки оформления заказа изображен на рисунке 11.

```
Mark
@GetMapping("/getcookies")
public String cookieGet(@RequestParam("cookies") String param, RedirectAttributes redirectAttributes)
{
    if (workshopService.convertCookies(param, redirectAttributes))
    {
        return "redirect:/order";
    }

    return "error";
}


Mark *
@GetMapping("/order")
public String orderGet(@ModelAttribute("items") CartItem[] items, Model model)
{
    model.addAttribute( attributeName: "products", workshopService.convertItemsToProduct(items));
    model.addAttribute( attributeName: "sum", workshopService.getSum(workshopService.convertItemsToProduct(items)));
    return "order";
}
```

Рисунок 11 – Контроллер для обработки заказа

После оформления заказа он добавляется к пользователю. Обработка данного процесса изображена на рисунках 12-13.

```
Mark
@GetMapping("/getproducts")
public String productsGet(@RequestParam("products") String param, RedirectAttributes redirectAttributes)
{
    if (workshopService.convertCookies(param, redirectAttributes))
    {
        return "redirect:/order/add";
    }

    return "error";
}

Mark
@GetMapping("/order/add")
public String orderPost(Principal principal, @ModelAttribute("items") CartItem[] items, HttpServletResponse response)
{
    if (orderService.addOrder(Arrays.stream(items).toList(), principal.getName()))
    {
        var deleteCookie = new Cookie( name: "cartItems",  value: null);
        deleteCookie.setMaxAge(0);
        response.addCookie(deleteCookie);

        return "redirect:/account";
    }

    return "redirect:/error";
}
```

Рисунок 12 – Контроллер для создания заказа

```
2 usages  Mark
public boolean convertCookies(String param, RedirectAttributes redirectAttributes)
{
    try
    {
        var jsonString = param.split( regex: "=")[1];
        var objectMapper = new ObjectMapper();
        var items = objectMapper.readValue(jsonString, CartItem[].class);
        redirectAttributes.addFlashAttribute( attributeName: "items", items);
        return true;
    }
    catch (Exception e)
    {
        e.printStackTrace();
        return false;
    }
}

2 usages  Mark
public List<CartItem> convertItemsToProduct(CartItem[] items)
{
    try
    {
        var products = new ArrayList<CartItem>();
        for (var item : items)
        {
            item.setCost(productRepository.getProductsByTitle(item.getProductName()).getCost());
            products.add(item);
        }

        return products;
    }
    catch (Exception e)
    {
        e.printStackTrace();
        return null;
    }
}
```

Рисунок 13 – Сервис для создания заказа

Для просмотра всех заказов пользователь может зайти в свою учетную запись, где будут выведены личные данные пользователя и список его заказов. Код для учетной записи изображен на рисунках 14-15.

```
Mark *
@GetMapping("/account")
public String accountPage(Principal principal, Model model) {
    model.addAttribute( attributeName: "user", userService.getUserByName(principal.getName()));
    return "account";
}
```

Рисунок 14 – Контроллер для учетной записи

```
public class UserService
{
    @Autowired
    private UserRepository userRepository;

    3 usages  Mark
    public User getUserByName(String userName) { return userRepository.getUserByName(userName); }

    1 usage  Mark *
    public void addOrder(Order order, String name)    {
        var user = getUserByName(name);
        user.getOrders().add(order);
        userRepository.save(user);
    }
}
```

Рисунок 15 – Сервис для учетной записи

Для доступа к данным из базы данных используются репозитории, как например на рисунке 16.

```
6 usages  Mark
public interface WorkshopRepository extends JpaRepository<Workshop, Integer>
{
    1 usage  Mark
    Workshop getWorkshopByProduct(Product product);
}
```

Рисунок 16 – Сервис для магазина

3.4 Тестирование приложения

Используя программу Postman и браузер, были протестированы несколько конечных точек приложения. Для начала протестирован система авторизации/регистрации аккаунта, рисунок 17-18.

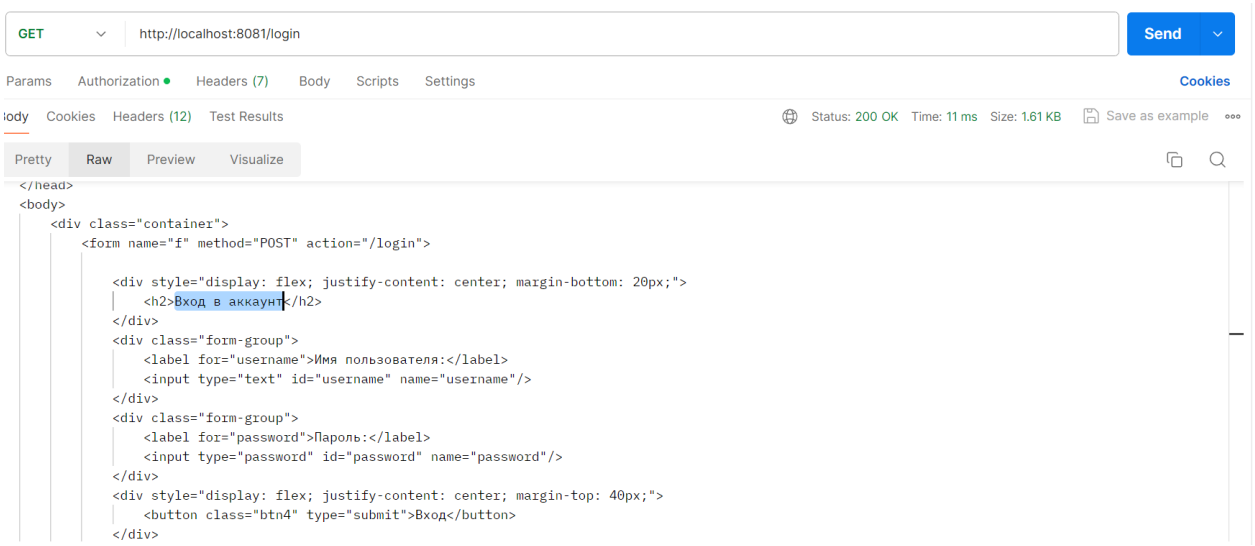

Рисунок 17 – GET запрос тестирования страницы авторизации

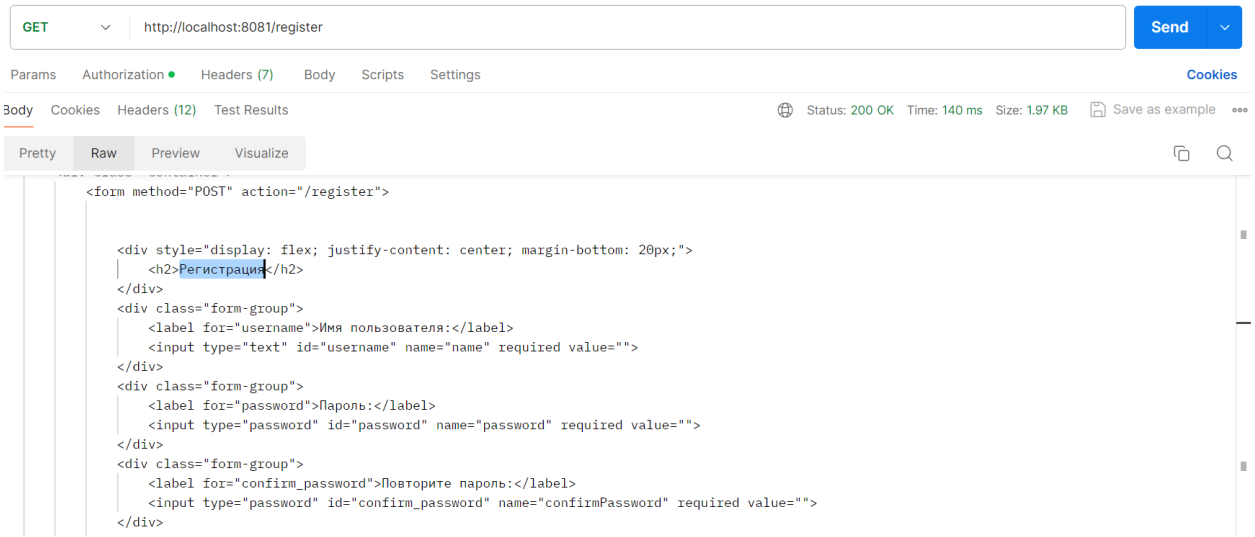

Рисунок 18 – GET запрос тестирования страницы регистрации

Далее, с помощью браузера, протестировано добавление товаров из каталога в корзину, рисунок 19.

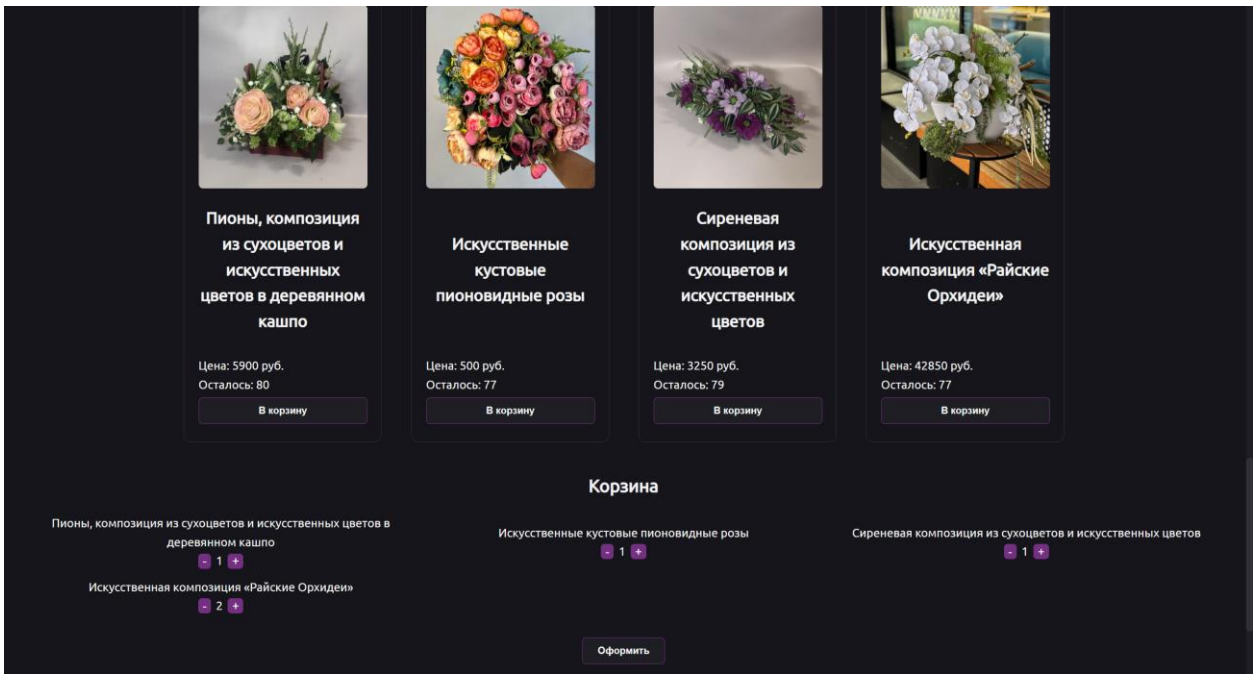

Рисунок 19 – Тестирование добавления товаров в корзину

Тестирование формирования заказа показано на рисунке 20.

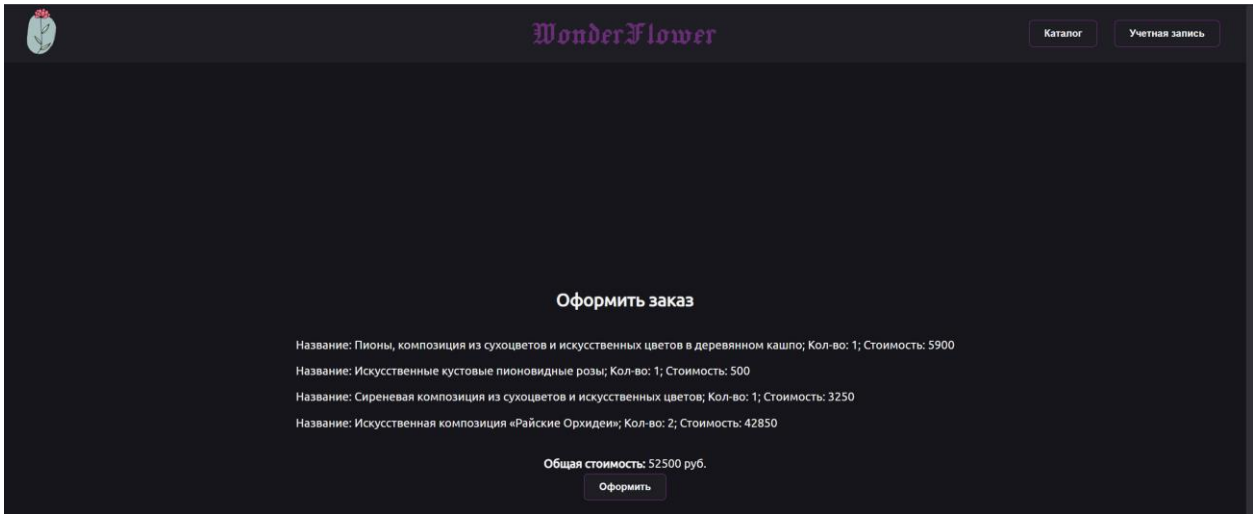

Рисунок 20 – Тестирование формирования заказа

При открытии страницы личного кабинета будет выведена информации об истории заказов, кнопка выхода из аккаунта и имя пользователя, рисунок 21.

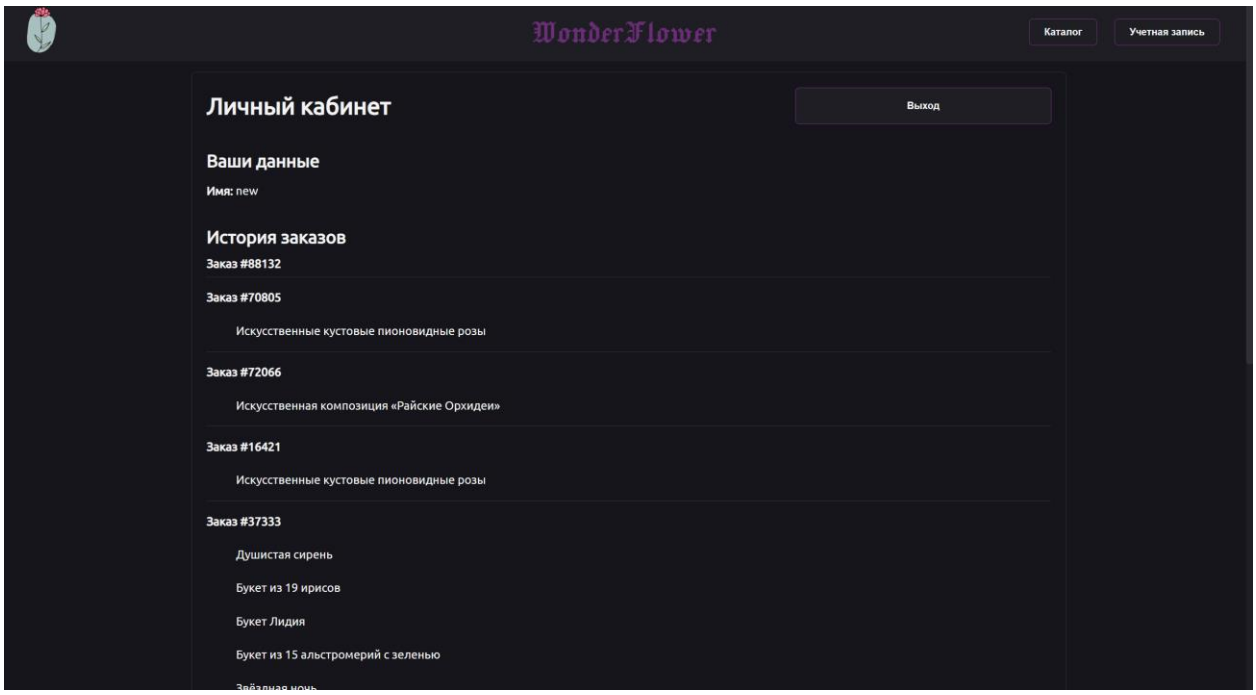

Рисунок 21 – Тестирование вывода заказов пользователя

3.5 Контейнеризация

Для развертывания на хостинге приложения, необходимо прописать Dockerfile, для создания контейнера, в котором будет работать разработанное приложения. В Dockerfile прописан основной образ, команда для копирования всех файлов в образ и запуск jar-файла. Содержимое Dockerfile показано на рисунке 22.

```
FROM openjdk:21-ea-21-jdk

WORKDIR /app

COPY . /app

EXPOSE 8081

ENTRYPOINT ["java","-jar","/app/build/libs/flowershop-0.0.1-SNAPSHOT.jar"]
```

Рисунок 22 – Файл Dockerfile

ЗАКЛЮЧЕНИЕ

В результате процесса разработки приложения были успешно достигнуты поставленные цели и выполнены задачи. Была проведена аналитика предметной области, определены оптимальные инструменты разработки, создано приложение с применением выбранных средств и протестировано его функционирование. Результатом этого процесса является стабильное приложение, которое обладает интуитивно понятным пользовательским интерфейсом и полным набором необходимых функций.

Результаты тестирования выявили высокую оценку по критериям корректности функционирования и оптимизации использования ресурсов. Благодаря своим качественным характеристикам, данное приложение обладает потенциалом для успешной конкуренции на рынке интернет-приложений с аналогичными по тематике сервисами. Выполненный отчет был проверен на антиплагиат, по данной курсовой работе была сделана презентация. Все поставленные задачи были выполнены.

Разработанное веб-приложение было развернуто на открытый хостинг от Render: https://javaflowershop.onrender.com, исходный код расположен на сайте GitHub: https://github.com/Shumila71/JavaFlowerShop.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. OpenJDK 21 Documentation [Электронный ресурс]. – URL: https://devdocs.io/openjdk~21/ (дата обращения 05.04.2024).

2. Spring Framework [Электронный ресурс]. – URL: https://spring-projects.ru/projects/spring-framework/ (дата обращения 12.04.2024).

3. PostgreSQL Documentation [Электронный ресурс]. – URL: https://www.postgresql.org/docs/ (дата обращения 12.04.2024).

4. Get started in Postman [Электронный ресурс]. – URL: https://learning.postman.com/docs/getting-started/overview/ (дата обращения 18.04.2024).

5. Руководство по Docker [Электронный ресурс]. – URL: https://my-js.org/docs/guide/docker/ (дата обращения 25.04.2024).

6. Flowwow [Электронный ресурс]. – URL: https://flowwow.com/moscow/ (дата обращения 12.04.2024).

7. Простоцветы [Электронный ресурс]. – URL: https://www.prostocvet.ru/ (дата обращения 12.04.2024).

8. Мега цветы [Электронный ресурс]. – URL: https://megacvet24.ru/ (дата обращения 13.04.2024).

9. Gradle User Manual [Электронный ресурс]. – URL: https://docs.gradle.org/current/userguide/userguide.html (дата обращения 11.04.2024).

10. Thymeleaf Documentation [Электронный ресурс]. – URL: https://www.thymeleaf.org/documentation.html (дата обращения 16.04.2024).

11. Render Quickstarts [Электронный ресурс]. – URL: https://docs.render.com/ (дата обращения 02.05.2024).