МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ОНЛАЙН ПОКУПКИ ЦВЕТОВ**

**Пояснительная записка**

Курсовой проект по дисциплине

«Web-программирование»

ТПЖА 090302.783 ПЗ

Разработал студент гр. ИТб-2301-03-24 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Овчарук А. И./

(подпись)

Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Земцов М. А./

(подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_» \_\_\_\_\_\_\_\_\_\_2023 г.

Киров 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра Систем автоматизации управления

**Утверждаю**

И.о. заведующего кафедрой САУ

\_ Ю.В. Ланских

« » 2023

**ЗАДАНИЕ**

на курсовую работу

по дисциплине *Web-программирование* Студент Овчарук Анна Игоревна Учебная группа ИТб-2301-03-24 Тема курсовой работы (проекта): Разработка веб-приложения онлайн покупки пиццы

Исходные данные:

1. *Описание автоматизируемых процессов:*
   1. *Каталог товаров цветочного магазина с возможностью добавить товар в корзину. Корзина может работать без авторизации.*
   2. *Наличие авторизации и регистрации клиента. Реализации функции покупки пиццы с эмуляцией функции оплаты и отправкой на почту товарного чека.*
   3. *Возможность просматривать заказы и при необходимости редактировать статус заказа.*
2. *Требования к используемым технологиям и языкам программирования:*
   1. *технологии реализации сервера приложений (backend): язык программирования*

***Java****, СУБД Postgresql*

* 1. *технологии реализации веб-интерфейса (frontend): язык программирования TypeScript, сервер Nodejs, JS framework - Angular 13, CSS framework - Tailwind CSS, Bootstrap 5, система сборки gulp, минимизатор webpac*

*3.Требования к инфраструктуре:*

*3.1. Исходный код всех проектов должен быть публично размещён в системе контроля версий git с указанием в пояснительной записке полного адреса к репозиторию. Использовать следующие публичные серверы для размещения исходного кода проектов: github.com, bitbucket.org, gitlab.com, gitflic.ru.*

*3.2. размещение данных только в предоставляемой бд Postgresql с доступом по логину и паролю:*

*4.Формат отчетных документов должен соответствовать требованиям ВятГУ*

Содержание пояснительной записки: *реферат, содержание, введение, описание предметной области и используемых технологий, разработка сервера приложений, разработка веб-интерфейса, заключение, библиографический список, исходный код ключевых файлов конфигурации, скрипты баз данных, руководство пользователя*

Графический материал: *структура веб-приложения, экранные формы.*

Руководитель работы *М.А. Земцов*

Подпись ФИО руководителя Дата

|  |  |  |  |
| --- | --- | --- | --- |
| Задание принял |  |  |  |
|  | Подпись | ФИО обучающегося |  |

**Реферат**

Овчарук А.И. Разработка веб-приложения онлайн покупки цветов: ТПЖА.090302.783 ПЗ: Курс. проект / ВятГУ, каф. САУ; рук. М.А. Земцов – Киров, 2023. ПЗ 39с., 29 рис., 4 табл., 2 прил.

ВЕБ-ПРИЛОЖЕНИЕ, FRONTEND, BACKEND, REST API, ANGULAR, TYPESCRIPT, JAVASCRIPT, JAVA, ENTITY FRAMEWORK, БАЗА ДАННЫХ, POSTGRESQL.

Объект исследования и разработки – веб-приложение онлайн покупки цветов.

Цель работы – разработка клиент-серверного веб-приложения онлайн покупки цветов с использованием фреймворка Angular и языка программирования Java.

В разработанном web-приложении авторизованный клиент может просматривать каталог товаров пиццерии с возможностью добавить товар в корзину и оформления заказа.

|  |  |  |  |
| --- | --- | --- | --- |
| **Содержание**  Изм.  Лист  № докум.  Подпись  Дата  Лист  2  ТПЖА.090302.783ПЗ  Разраб.  *Овчарук А.И.*  Провер.  Земцов М.А.  Реценз.  Н. Контр.  Утв.  Разработка веб-приложения онлайн покупки цветов  Литер  Листов  34  *Кафедра САУ,*  *группа ИТб-2301-03-24* | | |  |
| Введение………………………………………………………………..……… | | | 3 |
| 1 Описание предметной области и используемых технологий……………. | | | 4 |
| 1.1 Процессы подлежащие автоматизации………..…….……………..…. | | | 4 |
| 1.2 Описание используемых технологий…..………………………..…..... | | | 4 |
| 1.2.1Язык программирования Java, Gradle и фреймворка Micronaut………………………………………………………………. | | | 4 |
| 1.2.2 Управление базой данных с PostgreSQL и Hibernate …………. | | | 6 |
| 1.2.3 Преимущества TypeScript над JavaScript……………………….. | | | 6 |
| 1.2.4 Angular при работе с frontend……………………………………. | | | 7 |
| 1.2.5 Интеграция Bootstrap в проект Angular………………..……….. | | | 9 |
| 2 Разработка сервера приложения…………………………………………… | | | 9 |
| 2.1 Организация СУБД…………………………………………………….. | | | 9 |
| 2.2 Создание основных путей обращения к API…………………………. | | | 11 |
| 3 Разработка веб-интерфейса приложения…………………….…..………... | | | 17 |
| 3.1 Организация структуры проекта………………………..……………..  3.2 Основные компоненты Angular……………………………………….. | | | 17  18 |
| 3.3 Стилизация приложения………………………………………………..  3.4 Сценарный план взаимодействия…………………………………….. | | | 23  27 |
| Заключение…………………………………………………………………. | | | 25 |
|  | | |
|  |  |
| Приложение А (обязательное) Листинг программного кода реализации продуктов DTO…………………………………………………………….…. | | | 26 |
| Приложение Б (обязательное) Листинг кода маршрута отправки чека о покупке ………………………………………………………………………... | | | 28 |
|  | | |  |

# Введение

В век интернета наиболее востребованным ресурсом становится информация, которую человек может получить в неограниченном количестве, просто открыв браузер на своём компьютере или телефоне. Но чтобы эта информация была воспринята правильно, её необходимо структурировать и грамотно отобразить, поэтому в борьбе за целевую аудиторию предприниматель постоянно ищет удобные пути коммуникации с клиентом.

Задача курсовой работы заключается в разработке клиент-серверного веб-приложения онлайн покупки цветов с использованием фреймворка Angular, языка программирования Java и СУБД Postgresql для серверной части соответственно. В приложении должны быть реализованы основные и обязательные функции веб-магазина.

В первую очередь пользователь будет взаимодействовать с главной страницей приложения. На ней необходимо создать каталог товаров, где каждый сможет ознакомиться с информацией о цветах, а также, при нажатии на кнопку, добавить их в свою корзину. Выбранные позиции всегда отображаются в корзине с некоторой краткой информацией для удобства клиента. Данные функции должны быть доступны и для неавторизованного пользователя. Клиент может зарегистрироваться и авторизоваться. С серверной части приложение должно сохранить информацию о пользователе.

Оформление заказа происходит после подтверждения пользователем состава его корзины. На форме заказа клиенту необходимо ввести данные о себе и данные для самого заказа. Он может выбрать дату заказа и способ доставки. Сразу же после оформления клиент получит результат в всплывающем окне, а также письмо с информацией о заказе на свою почту, указанную при оформлении заказа. Со стороны серверной части заказ сохраниться в базе данных и будет указан на странице менеджера.

На странице менеджера будет реализован список заказов с полной информацией. Менеджер сможет изменить статус заказа или удалить его. У менеджера будет возможность просмотреть информацию о зарегистрированных пользователях. Также у менеджера будет реализован доступ к созданию заявки на добавление нового товара действующему программисту предприятия. После успешного создания заявки будет приходить письмо с информацией о товаре.

# 1 Описание предметной области и используемых технологий

Цветочный магазин – это предприятие, предоставляющее ассортимент всевозможных букетов и цветочных композиций. Учитывая, что цветы являются нестандартным товаром, существует необходимость связи между клиентом и менеджером-флористом.

Информационная система флористического салона предполагает хранение больших объёмов информации о товаре, о клиентах и их заказах. Бизнес-процессы магазина нуждаются в автоматизации.

# 1.1 Процессы, подлежащие автоматизации

Веб-приложение предназначено для оформления заказа клиента. Главная задача такого приложения – обеспечить удобство пользователю при заказе цветов.

Для ознакомления с имеющимся ассортиментом на стартовой странице располагается каталог, где можно посмотреть описание, изображение и цену товара. Чтобы ресурс был комфортным для пользователя, веб-приложению необходимо отличаться приятным дизайном и интуитивно понятным интерфейсом.

Клиент должен иметь возможность добавить выбранный товар в корзину и оформить заказ. Регистрация и авторизация клиента на сайте потребуются для сбора информационных данных, для сохранения в корзине выбранных букетов или композиций, а также для оформления заказа и дальнейшей эмуляции функции оплаты.

# 1.2 Описание используемых технологий

В разработке программного обеспечения термины «frontend» и «backend» относятся к разделению задач между уровнем представления и уровнем доступа к данным части программного обеспечения. В клиент-серверной модели клиент обычно считается внешним интерфейсом, а сервер обычно считается бэкендом. В текущем разделе приводится обзор задействованных средств, фреймворков и библиотек, необходимых для воплощения в жизнь серверной части веб-приложения и веб-интерфейса клиента.

# 1.2.1 Язык программирования Java, Gradle и фреймворка Micronaut

Backend включает в себя код, который используется на сервере, отвечающий за функционирование внутренней части веб-приложения. Под этим подразумевается взаимодействие с базой данных, API, обработка входящих данных, соединение различных веб-ресурсов между собой и так далее.

Наиболее популярными языками программирования при разработке серверной части являются языки JavaScript, Java, PHP и C#. Стоит отметить, что в чистом виде языки используются довольно редко – чаще используются фреймворки и библиотеки, которые содержат в себе уже готовые блоки кода, реализующие тот или иной функционал, позволяя тем самым сэкономить разработчику время на написание кода.

Java – Строго типизированный объектно-ориентированный язык программирования общего назначения. Это быстрый, безопасный и надежный язык программирования для всего: от мобильных приложений и корпоративного ПО до приложений для работы с большими данными и серверных технологий. Именно на этом языке с использованием различных фреймворков и библиотек будет написана серверная часть приложения.

Gradle — система для автоматизации сборки приложений и сбора статистики об использовании программных библиотек, применяющая языки Groovy, Java, JavaScript, Kotlin и т. д., а также решения из фреймворков Apache Ant и Apache Maven.

Клиентская часть приложения и сервер взаимодействуют друг с другом с помощью HTTP-запросов с использованием архитектуры REST API, которая использует следующие методы: GET (получение данных), POST (добавление или замена данных), PUT (обновление данных) и DELETE (удаление данных). Другими словами, REST – это набор правил указывающих, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было легко масштабировать. REST API позволяет обмениваться не только текстовой информацией, но и данными в специальных форматах: JSON и XML.

Micronaut — это фреймворк на JVM(Java Virtual Machine) для построения легковесных модульных приложений. Micronaut это современный фреймворк, призванный сделать создание микросервисных приложений быстрым и простым. Micronaut содержит возможности похожие на существующие фреймворки, такие как Spring, но в то же время он реализует некоторые новые идеи, которые являются его отличительными чертами. Вместе с поддержкой Java, Groovy и Kotlin он предлагает множество путей создания приложений.

Одна из самых захватывающих возможностей Micronaut это внедрение зависимостей (DI) во время компиляции. Micronaut собирает данные для внедрения зависимостей на этапе компиляции. Результат этого — быстрое время старта приложений и меньшее потребление памяти. Еще одна его возможность — это первоклассная поддержка реактивного программирования, как для клиентов, так и для серверов.

# 1.2.2 Управление базой данных с PostgreSQL и Hibernate

PostgreSQL не просто реляционная, а объектно-реляционная система управления базой данных, которая базируется на языке SQL. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird. Фундаментальная характеристика объектно-реляционной базы данных – это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает Postgres невероятно гибким и надежным. Среди прочего он умеет создавать, хранить и извлекать сложные структуры данных.

Hibernate — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (ORM). Позволяет сократить объёмы низкоуровневого программирования при работе с реляционными базами данных; может использоваться как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей базой.

Библиотека не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

# 1.2.3 Преимущества TypeScript над JavaScript

Frontend – это публичная часть web-приложений, с которой пользователь может взаимодействовать и контактировать напрямую. Во Frontend входит отображение функциональных задач, пользовательского интерфейса, выполняемых на стороне клиента, а также обработка и отправка на сервер пользовательских запросов. Основными составляющими фронтенд-разработки являются HTML, CSS и JavaScript.

JavaScript – это мультипарадигменный язык программирования, обычно использующийся как встраиваемый для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах в качестве языка сценариев для придания интерактивности веб-страницам. JavaScript может обновлять как HTML, так и CSS, его задача – откликаться на действия пользователя.

Но для больших приложение возможностей JavaScript бывает недостаточно. Например, в JavaScript используется динамическая типизация, то есть тип переменных может меняться в процессе выполнения кода, что при большой кодовой базе и нескольких разработчиках часто становится источником ошибок. С начала 2010-х программисты искали инструмент, который помог бы справиться с этими недостатками. Одним из таких инструментов и стал TypeScript.

Для того, чтобы обойти ограничения JavaScript разработчики TypeScript изменили типизацию с динамической на статическую, расширили возможности объектно-ориентированного программирования и добавили обратную совместимость с JS. TypeScript предоставляет разработчику возможность работать с классами, интерфейсами, пространствами имён, наследованиями, интерфейсами, комментариями, выражениями и модулями.

Поддержка использования объектно-ориентированного подхода, стрелочных функций, подключения модулей и других преимуществ TypeScript призваны повысить скорость разработки, облегчить читаемость, повторное использование кода и помочь осуществлять поиск ошибок на этапе разработки и компиляции.

# 1.2.4 Angular при работе с frontend

В условиях динамически развивающихся технологий появляется всё больше различных библиотек и фреймворков, ориентированных на предоставление пользователю удобного интерфейса и упрощение создания технически сложных или нагруженных проектов. Правильно подобранная среда может значительно облегчить работу: программисту придется писать меньше кода вручную, потому что уже есть заранее написанные и готовые к использованию функции и шаблоны.

Angular – это кроссплатформенный фреймворк, разработанный компанией Google, который придерживается строгой иерархии и представляет собой большую инфраструктуру для комплексных решений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений.

Статически типизированный язык TypeScript является прекрасным дополнением JavaScript и незаменим при построении сайтов. Одной из ключевых особенностей Angular является встроенный TypeScript, весьма улучшающий инструментарий фреймворка.

Структуру frontend условно можно поделить на какое-то определённое количество блоков, каждый из которых будет иметь свою реализацию. Такие блоки можно назвать компонентами – составные части приложения со своей логикой, шаблоном HTML и CSS-стилями. Например, один компонент – это лента новостей, другой – шапка сайта. При этом Angular допускает возможность повторного их использования, что особенно полезно в корпоративных приложениях, где разные системы сходятся воедино, но могут иметь много похожих элементов. Приложение строится из них, как из блоков. Angular помогает привязывать компоненты друг к другу и передавать между ними данные.

Почти все современные фреймворки используют парадигму компонентного подхода, и Angular далеко не первый, кто предлагает такое решение. Но если сравнивать подход Angular с подходами других фреймворков, то можно выявить ряд отличий: здесь можно не только создавать новые теги для HTML-разметки, но и изменять существующие. В Angular можно вынести вёрстку отдельного блока в отдельный файл, что существенно упрощает работу верстальщика. Фреймворк, основанный на компонентном подходе, позволяет легче масштабировать и обновлять приложение.

Сервисы в Angular представляют довольно широкий спектр классов, которые выполняют некоторые специфические задачи, например, логгирование, работу с данными и так далее. В отличие от компонентов и директив сервисы не работают с представлениями, то есть с разметкой html, не оказывают на нее прямого влияния. Сервисы в Angular могут использоваться для хранения глобального состояния приложения, а также могут предоставить канал взаимодействия между отдельными компонентами.

Сервис может инкапсулировать бизнес-логику, различные вычислительные задачи, задачи по логгированию, которые лучше выносить из компонентов. Тем самым код компонентов будет сосредоточен непосредственно на работе с представлением. Кроме того, тем самым мы также можем решить проблему повторения кода, если нам потребуется выполнить одну и ту же задачу в разных компонентах и классах.

Angular содержит все то необходимое, что может упростить разработку крупномасштабных интерфейсов.

# 1.2.5 Интеграция Bootstrap в Angular

Angular представляет собой набор хорошо интегрированных библиотек, охватывающих широкий спектр функций: маршрутизация, управление формами, клиент-серверное взаимодействие, анимации и так далее.

NG-Bootstrap – это библиотека UI-компонентов, написанная на языке Typescript, которая открывает внутри Angular все возможности фреймворка Bootstrap.

**Bootstrap** – это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для **быстрого создания адаптивных дизайнов сайтов**. Фреймворк Bootstrap используется не только независимыми разработчиками, но и **целыми компаниями**. Основная область его применения – это разработка фронтен.д составляющих сайтов и интерфейсов.

В сущности, Bootstrap – это **просто набор файлов** (CSS и JavaScript), после подключения которых странице станут доступны **большое количество классов и готовых компонентов** для верстки дизайна**. Это позволяет верстать сайты в несколько раз быстрее, чем на «чистом» CSS** и JavaScript. **Оформление HTML-элементов и компонентов Bootstrap выполнено в едином стиле в последних тенденциях веб-дизайна.**

# 2 Разработка сервера приложения

В рамках курсовой работы разработка сервера приложения заключается в организации процесса программного управления базой данных и написания основных маршрутов обращения запросов клиента.

# 2.1 Организация СУБД

СУБД обеспечивает обработку и хранение всех данных магазина. В обозначенной предметной области фигурируют следующие классы: клиент, заказ клиента, корзина клиента, товар, заказ, оплата. Используемые классы и их отношения между друг другом представлены в диаграмме на рисунке 2.

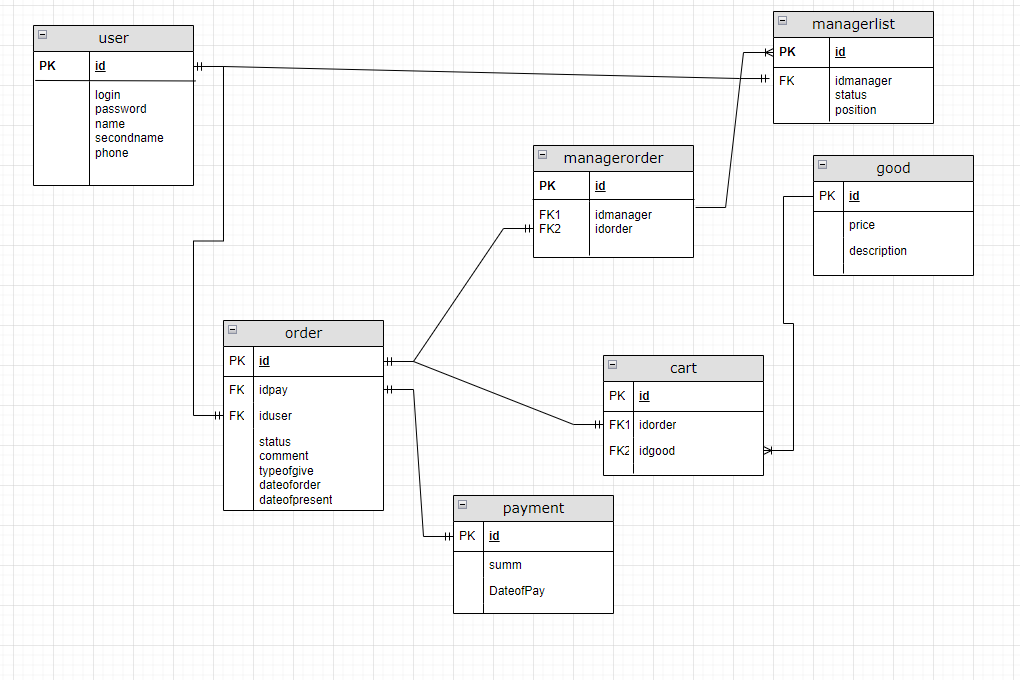


Рисунок 2 – Диаграмма классов UML

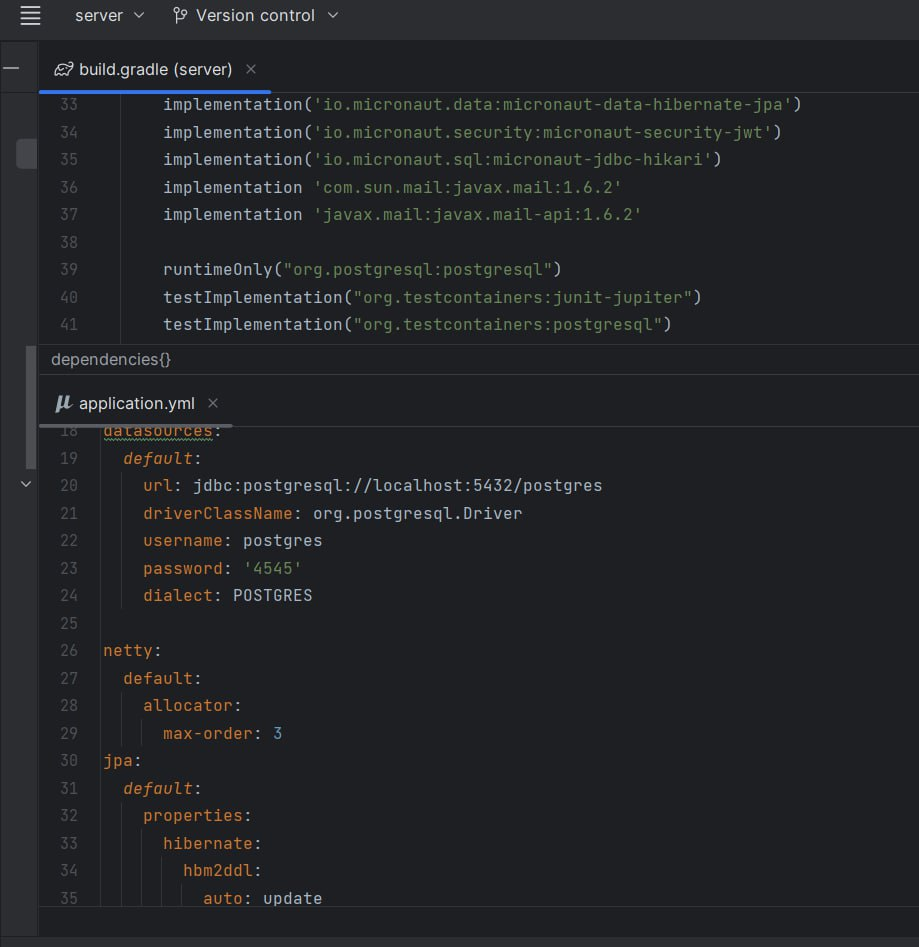
Взаимодействие с базой данных происходит через подключенную зависимость в файле build.gradle и сами данные подключения в файле свойств application.yml. Программная реализация приведена на рисунке 3.

Рисунок 3 – Программная реализация

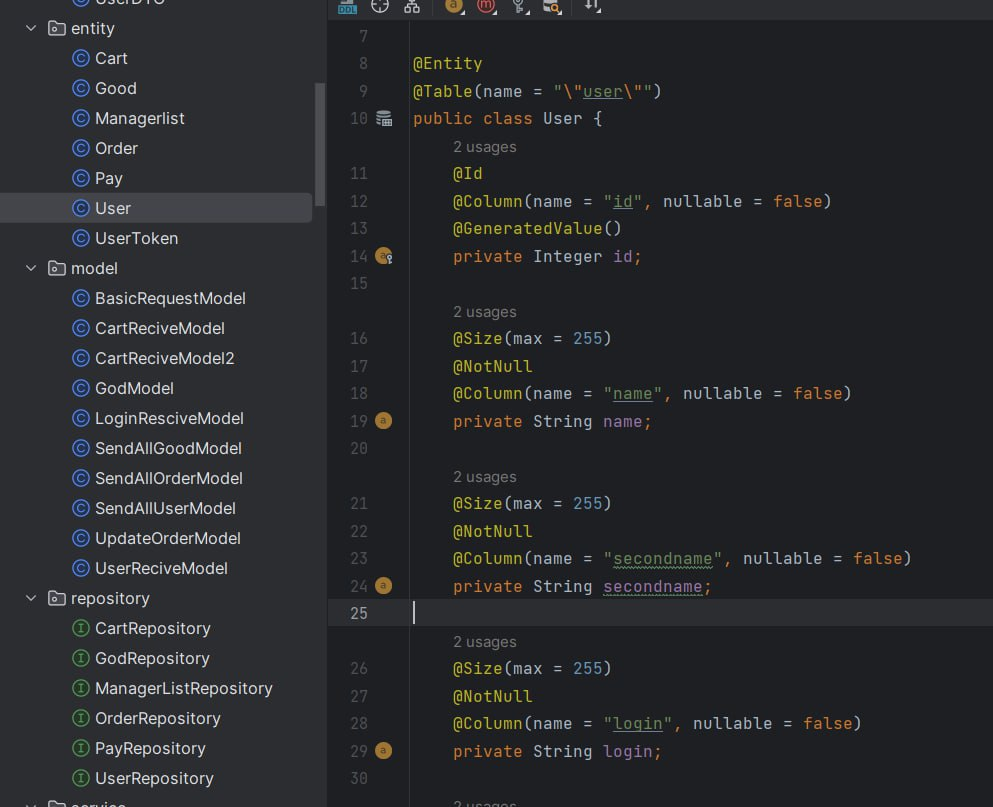
Для таблиц, их колонок и связей между ними были прописаны сущности, их атрибуты и ключи в пакете entity. Пример программного кода для сущности клиента приведен на рисунке 4.

Рисунок 4 – Пример программного кода для сущности клиента

В этом разделе продемонстрирована диаграмма UML с отображением классов и связей между ними и сгенерированы сущности в базе данных.

# 2.2 Создание основных путей обращения к API

Один из самых важных архитектурных принципов заключается в уникальности адресов ресурсов. Маршрут – это адрес, по которому отправляется запрос клиента. Каждая единица данных должна иметь свой собственный уникальный URL, который, по сути, целиком является однозначным идентификатором ресурса.

Программный код get-запросов получения всех товаров написан в контроллерах. Основная функция контроллера — это вызывать методы у нужных объектов, управлять доступом к ресурсам для выполнения задач, заданных пользователем. Программный код get-запросов получения всех товаров, списка клиентов их заказов из базы данных приведён на рисунках 5–7 соответственно.

При проектировании информационной системы, подразумевающей активный обмен данными, выбор формата сериализации является одной из первостепенных задач, существенно влияющей на эффективность сжатия и скорость чтения информации. JSON – это текстовый формат обмена данными, основанный на подмножестве языка JavaScript. JSON легко читается людьми, в качестве значений могут использоваться числа, строки, массивы, JSON-объекты и литералы (логические значения true, false и null). Сериализация означает преобразование объекта в строку, а десериализация – обратная операция. В рамках курсовой работы формат JSON будет использоваться для обмена данными между клиентской стороной и серверной.

Рисунок 5 – Программный код маршрута, предоставляющего клиентов из базы данных

Рисунок 6 – Программный код маршрута, предоставляющего товары из базы данных



Рисунок 7 – Программный код маршрута, предоставляющего заказы из базы данных

Для сериализация и десериализации данных были созданы специальные классы, которые в последующем используются для работы с базой данных. Пример сериализация и десериализации данных товара представлен на рисунках 8-9.

Data Transfer Object (DTO) — один из шаблонов проектирования, используется для передачи данных между подсистемами приложения. Содержит механизмы сериализации и десериализации. В DTO мы можем хранить данные из одного источника или из нескольких ресурсов. Можем хранить либо полные данные, либо небольшой объем данных из источника. DTO товара приведено на рисунке 10.

Также были созданы репозитории для каждой сущности, которые расширяют CrudRepository – интерфейс для общих операций CRUD (создание, чтение, обновление и удаление).

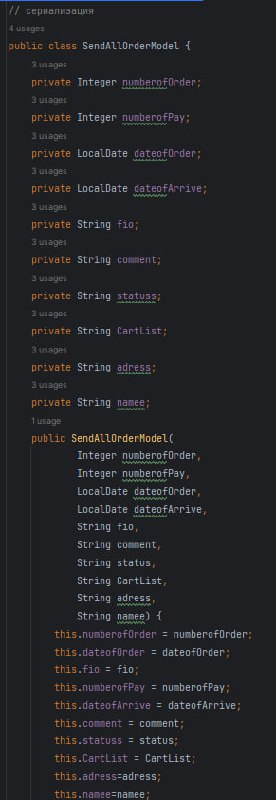


Рисунок 8 – Пример сериализация данных товара

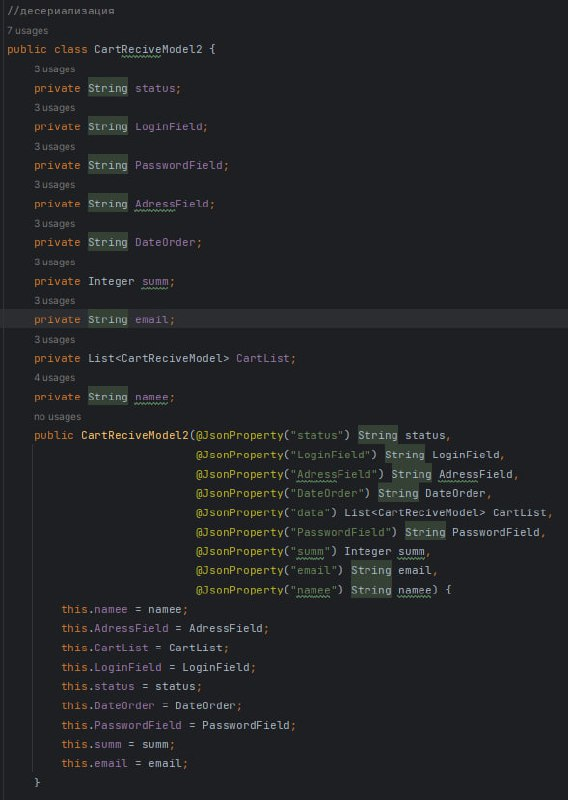
Рисунок 9 – Пример десериализация данных товара

Рисунок 10 – Продукты DTO

Листинг программного кода реализации продуктов DTO приведен в приложении А.

Основная бизнес-логика прописана в классах Service.

Для отправки почты используется протокол SMTP (Simple Mail Transfer Protocol). Данный протокол указывает, как почтовые сервера взаимодействуют при передаче электронной почты, он не рассчитан на обработку входящих сообщений, его используют для отправки и последующей доставки писем адресату. Преимущественно с помощью SMTP отправляют массовые и транзакционные рассылки.

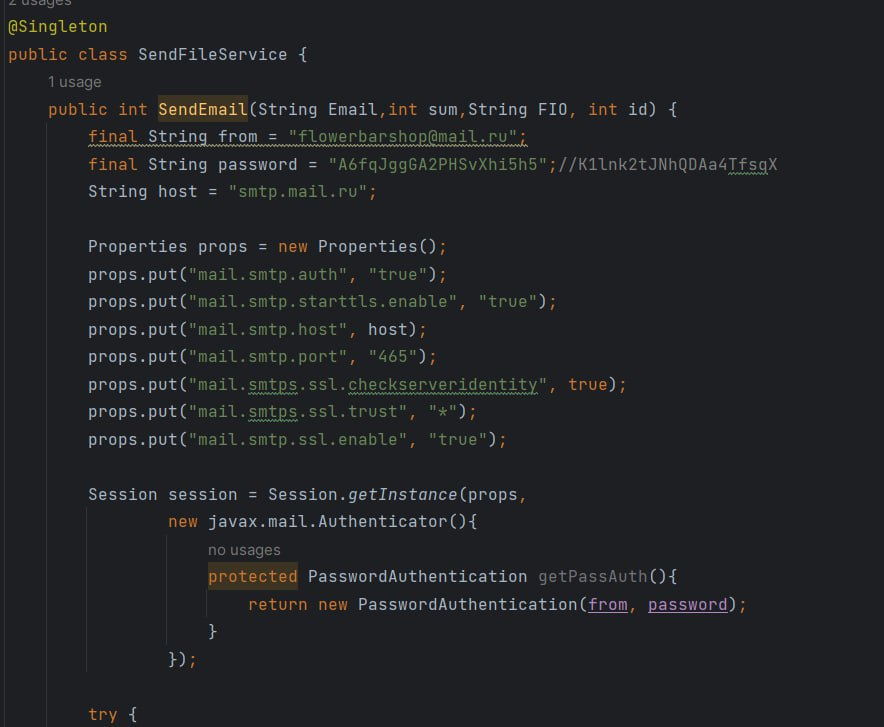
Для работы с протоколом SMTP и отправки электронной почты необходимо было подключить зависимости в build.gradle, и , предварительно создав почту mail и внешний ключ к ней, в классе SendFileService импортировать JavaMail API. Он предоставляет классы, которые моделируют почтовую систему. Пакет javax.mail определяет классы, которые являются общими для всех почтовых систем. Программное подключение приведено на рисунке 11.

Рисунок 11 – Программное подключение

Клиент получает на почту, указанную при регистрации, отчёт о конкретной покупке. В приложении Б демонстрируется листинг кода маршрута, выполняя запрос к которому, пользователю на почту отправляется список совершённого им заказа.

В текущем разделе приведён программный синтаксис основных маршрутов обращения к серверу и используемых средств для их построения.

# 3 Разработка веб-интерфейса

Веб-приложения должно быть таким, чтобы пользователь без труда находил нужную информацию, не теряясь в функционале и многочисленных страницах, и при этом получая удовольствие от работы с ресурсом. Грамотно продуманный «внешний вид» сайта – залог доверия со стороны потребителя, попавшего на сайт впервые.

# 3.1 Структура проекта

Прототипирование сайта – это процесс создания макета веб-страницы, наброска будущего сайта. Прототип помогает продумать структуру, не углубляясь в дизайн. Обычно прототип выглядит как схематичное изображение страницы, которое показывает, какие есть блоки и сущности.

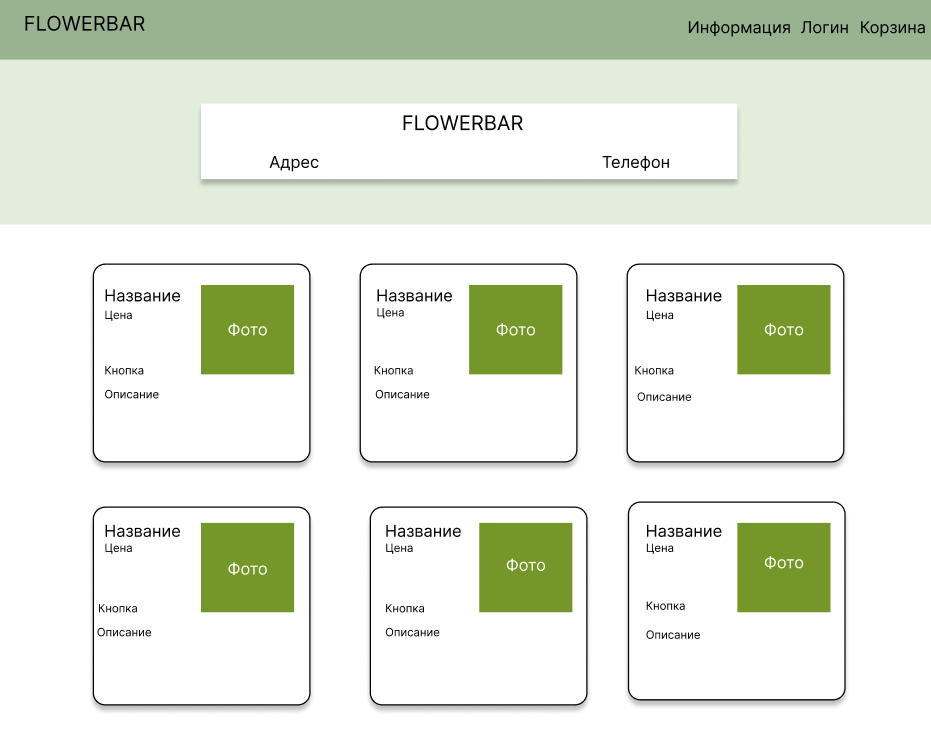
Главная страница веб-приложения представляет собой страницу каталога товаров. Здесь клиент знакомится с ассортиментом, он может добавить товар в корзину и посмотреть информацию об этапах заказа. Прототип страницы каталога товаров, построенный в графическом редакторе, представлен на рисунке 12.

Рисунок 12 – Прототип страницы каталога с товарами

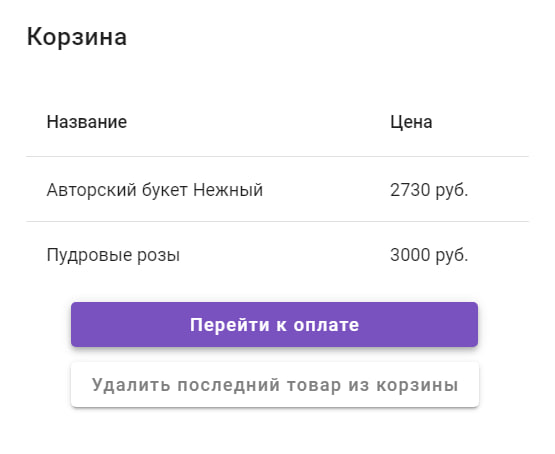
Вторая по важности часть веб-приложения – это корзина. Здесь клиент может ознакомиться со списком выбранных блюд и отсюда перейти на вкладку оформления заказа. Прототип всплывающего окна с корзиной веб-приложения, построенный в графическом редакторе, представлен на рисунке 13.

Рисунок 13 – Прототип корзины веб-приложения

В разделе продемонстрирован прототип нескольких страниц веб-приложения и показаны визуальные решения с сайтов со схожей тематикой, которые будут взяты за основу веб-интерфейса.

# 3.2 Основные компоненты Angular

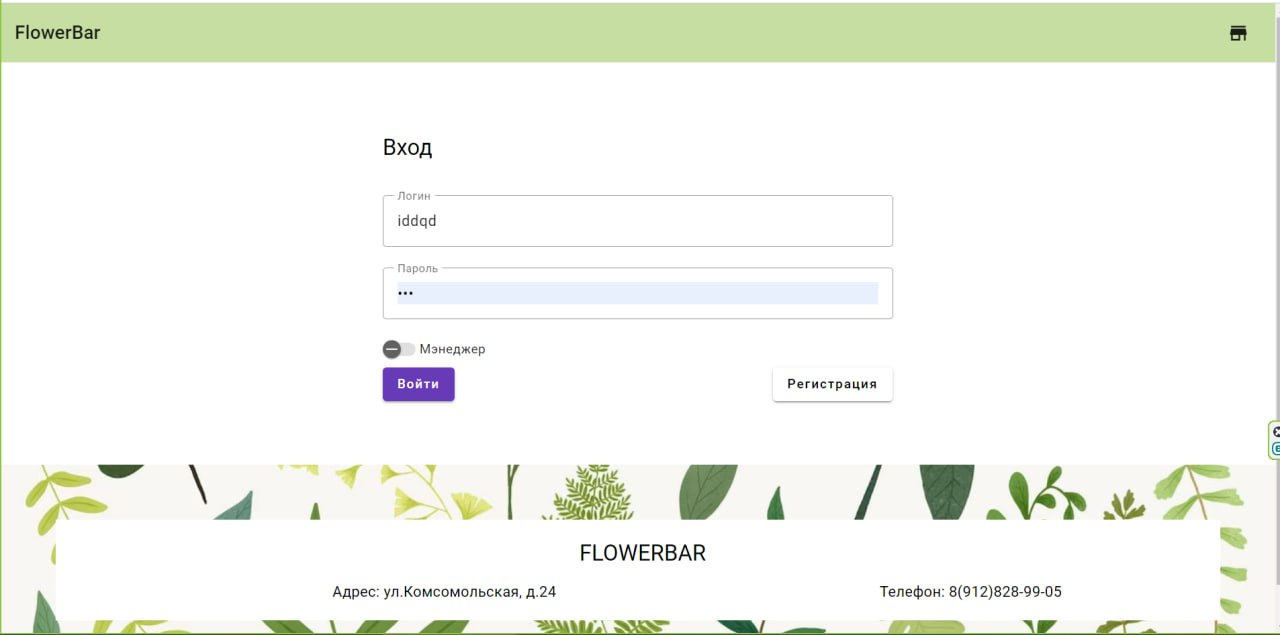
Компоненты – это основные программные инструменты создания сайта.  Они отображают данные и делят пользовательский интерфейс на меньшие представления, которые допускают многократное использование.

Навигационная панель, необходимая для перемещения по веб-приложению, есть на каждой странице. Из любой точки сайта можно перейти на страницу с каталогом блюд, на страницу входа или в корзину. На рисунке 14 продемонстрировано навигационное меню.



Рисунок 14 – Навигационное меню

В веб-приложении используются формы для регистрации клиента. Регистрация и авторизация обязательны для оформления заказа. На рисунке 15 изображен компонент формы входа, а на рисунке 16 – пример обработки входных данных при вводе неподходящего значения.



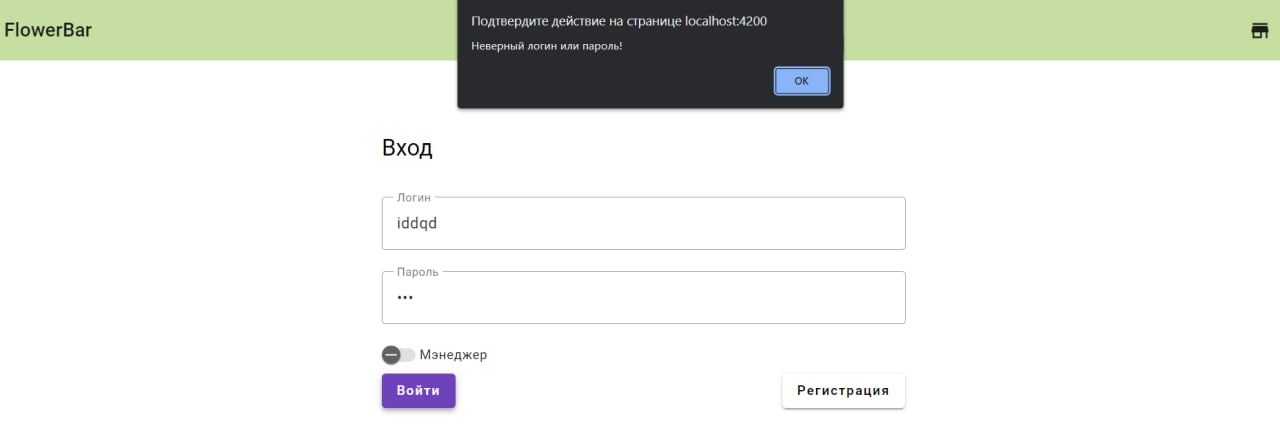
Рисунок 15 – Форма входа в аккаунт

Рисунок 16 – Пример обработки значений формы входа

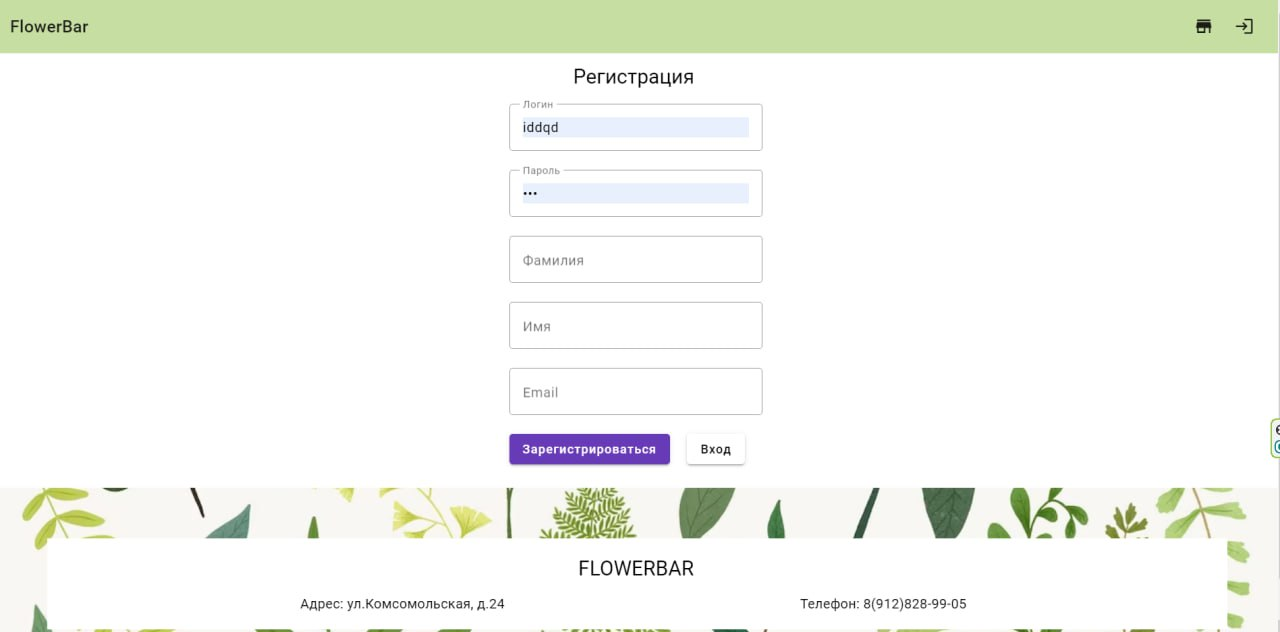
На рисунке 17 продемонстрирован компонент регистрации нового клиента с обрабатываемым входным значением.

Рисунок 17 – Форма регистрации клиента

Модальное окно – это элемент интерфейса, визуально представляющий собой «всплывающее окно», которое отображается над остальной частью страницы. Модальное окно в веб-приложении сопровождается затемнением остальной части страницы. Это действие позволяет визуально отделить его от содержимого страницы и показать, что в данный момент только оно одно является активным элементом. На рисунке 18 показано модальное окно оформления заказов.

Рисунок 18 – Модальное окно оформления заказов

В целях сохранения единого сценария взаимодействия клиента с веб-приложением было принято решение реализовать модальные окна и для менеджера для просмотра списка пользователей (рисунок 19).

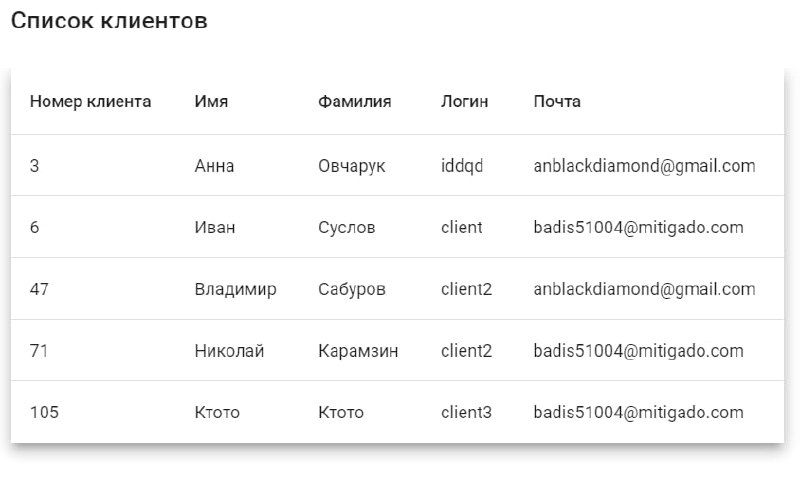


Рисунок 19 – Модальное окно просмотра списка пользователей

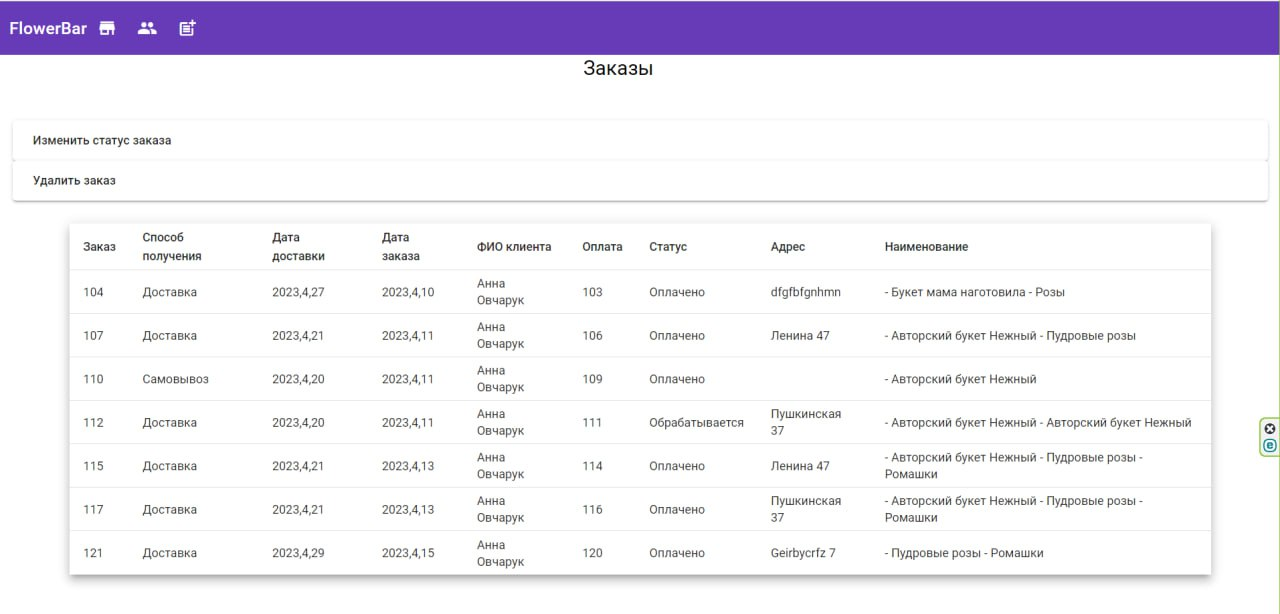
На рисунке 20 показана страница менеджера, где пользователь может изменить данные заказа, посмотреть историю, список пользователей и сделать заявку на добавление нового заказа. Отсюда клиент может перейти во вкладку с магазином.

Рисунок 20 – Страница менеджера

Сразу после оформления заказа обновятся данные истории заказов в кабинете менеджера, и отправиться письмо с подтверждением о покупке, на рисунке 21 продемонстрирован ответ сервера, отправляющего отчёт на почту.

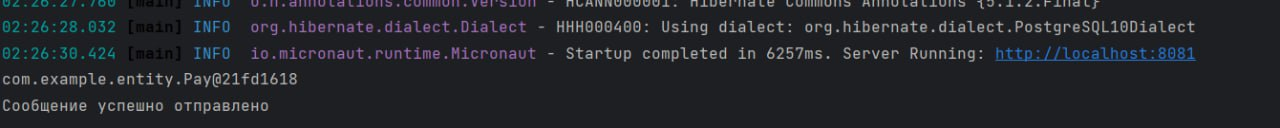


Рисунок 21 – Ответ сервера на отправку отчёта о покупке на почту

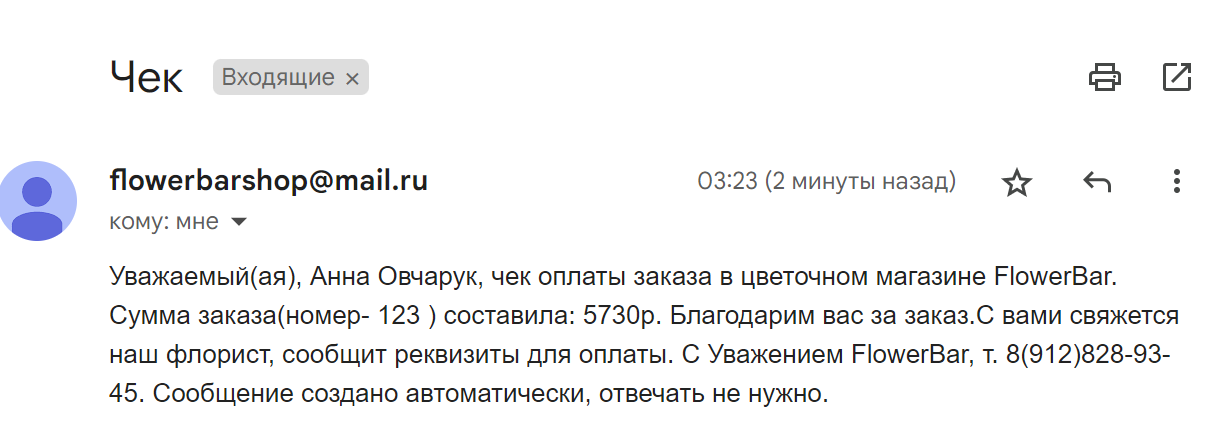
На почту, указанную при регистрации, придёт письмо, фрагмент которого приведён на рисунке 22.

Рисунок 22 – Фрагмент электронного письма

Сразу после добавления заявки на новый товар отправиться письмо с подтверждением, на рисунке 23 продемонстрирован ответ сервера, отправляющего отчёт на почту.

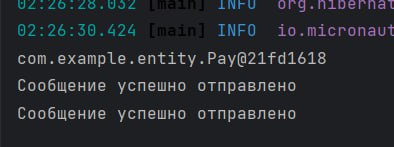


Рисунок 23 – Ответ сервера на отправку отчёта о покупке на почту

На почту магазина придёт письмо, фрагмент которого приведён на рисунке 24.

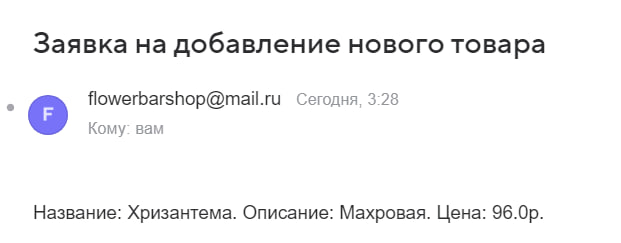


Рисунок 24 – Фрагмент электронного письма

В разделе продемонстрированы реализованные программные компоненты веб-приложения.

# 3.3 Стилизация приложения

При создании веб-сайта прежде всего необходимо применять знания психологии цвета и современных тенденций оформления. Важно, чтобы дизайн соответствовал общепринятым нормам, был приятен глазу, не отталкивал, а, наоборот, привлекал внимание. Стилизация приложения выполнена на основе трехцветной гаммы, которая вместе со сдержанностью в оформлении создаёт симпатичный и не перегруженный внешний вид. В качестве основных цветов были выбраны: теплый оливковый, белый, фиолетовый.

Выбор оливкого цвета продиктован желанием отличиться от других сайтов. Зелёный цвет ассоциируется в первую очередь с природой, спокойствием и ввиду своей естественности вызывает доверие пользователя. Контрастный зелёному фиолетовый используется для управления вниманием пользователя, для подчеркивания разницы между элементами сайта. А белый– для оформления свободного пространства, служит фоном веб-приложения, обеспечивает удобную читаемость заголовков и текста. Кнопки и ссылки веб-приложения стилизованы одинаково. На рисунке 25 представлен скриншот экрана главной страницы.

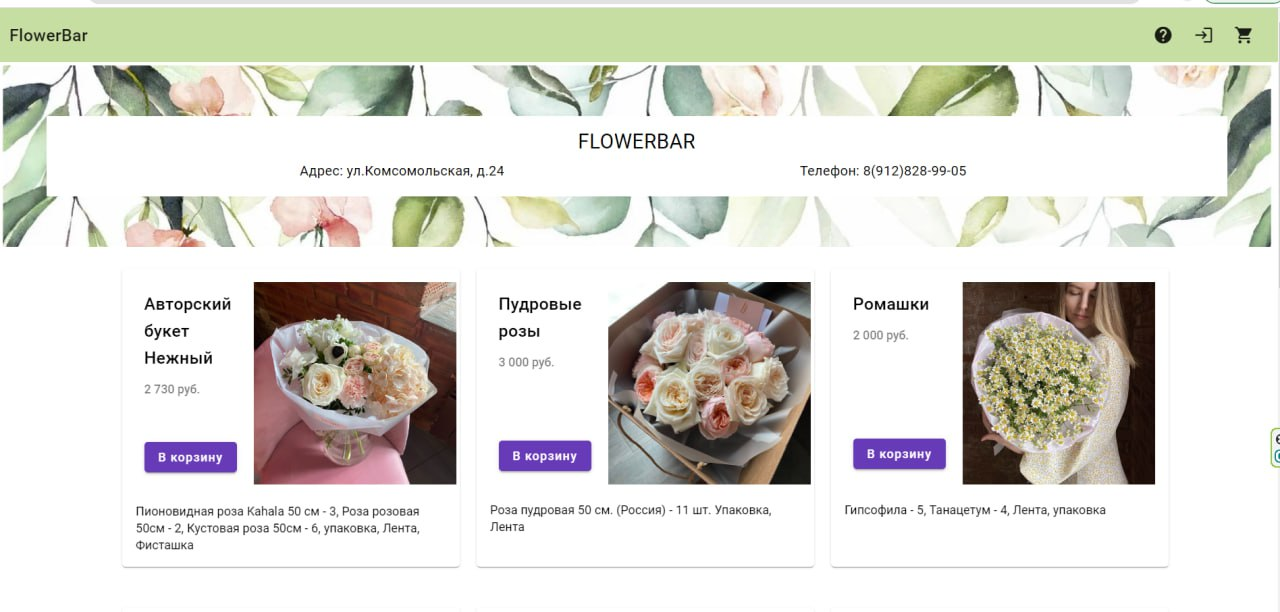
****

Рисунок 25 – Скриншот экрана главной страницы

# 3.4 Сценарный план взаимодействия пользователя с программным продуктом

Use Case — это сценарный план взаимодействия пользователя с программным продуктом, в котором четко прописаны шаги для достижения того или иного результата.

Этапы работы со страницей регистрации прописаны в таблице 1. Страница регистрации представлена на рисунке 26.

Таблица 1 - Этапы работы со страницей регистрации

|  |  |  |
| --- | --- | --- |
| Номер шага | Действующее лицо | Действие |
| 1 | Пользователь | Пользователь нажимает на кнопку регистрации |
| 2 | Система | Открывает форму регистрации |
| 3 | Пользователь | Пользователь заполняет форму, подтверждает регистрацию |
| 4 | Система | Пользователь вносится в базу данных |
| 5 | Пользователь | Получает сообщение о результате регистрации |

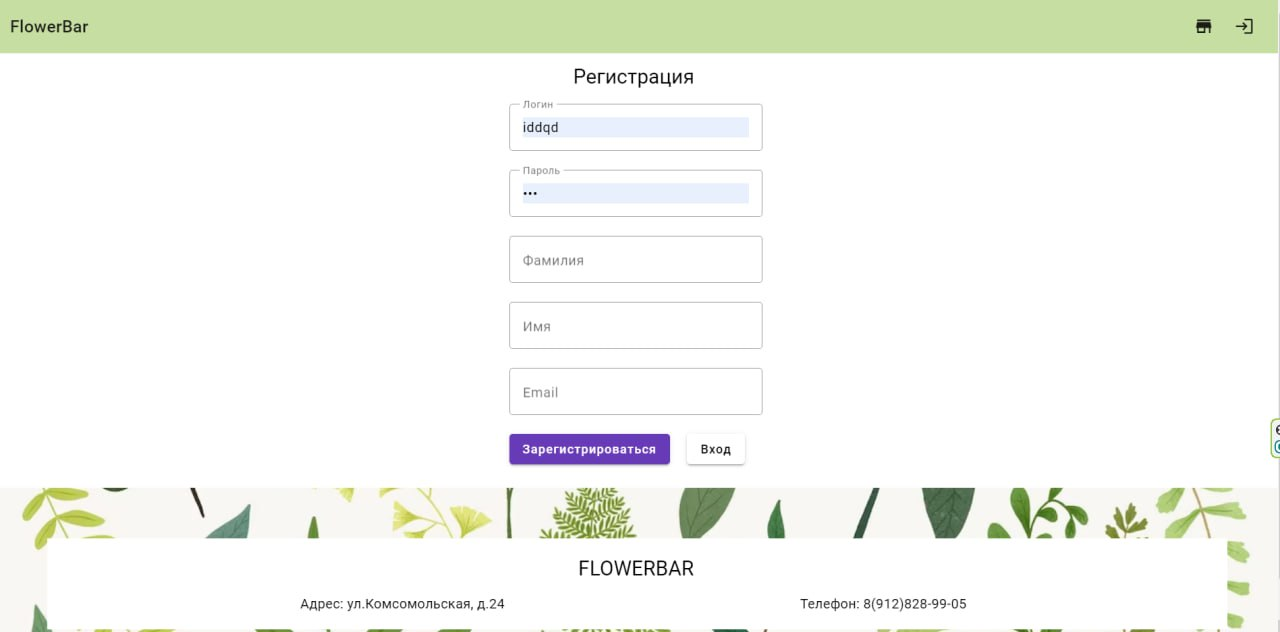
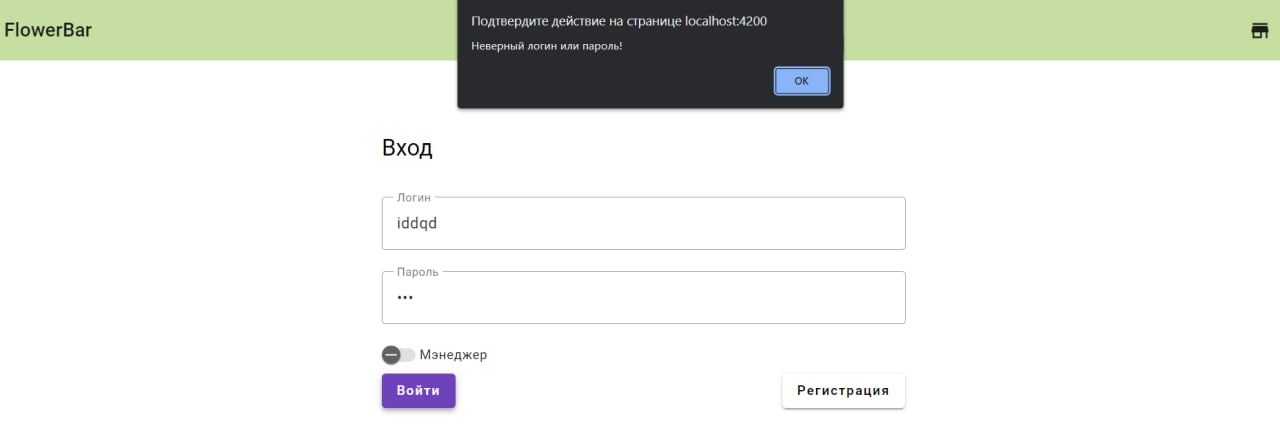


Рисунок 26 – Страница регистрации

Этапы работы со страницей входа прописаны в таблице 2. Страница входа представлена на рисунке 27.

Таблица 2 - Этапы работы со страницей входа

|  |  |  |
| --- | --- | --- |
| Номер шага | Действующее лицо | Действие |
| 1 | Пользователь | Пользователь нажимает на кнопку входа |
| 2 | Система | Открывает форму входа |
| 3 | Пользователь | Пользователь заполняет форму, подтверждает вход |
| 4 | Система | Сравнивает с данными в базе данных и выводит соответствующее сообщение |
| 5 | Пользователь | Получает сообщение о результате входа |

Рисунок 27 – Страница входа

Этапы работы с главной страницей прописаны в таблице 3. Страница регистрации представлена на рисунке 28.

Таблица 3 - Этапы работы с главной страницей

|  |  |  |
| --- | --- | --- |
| Номер шага | Действующее лицо | Действие |
| 1 | Пользователь | Просматривает список товаров. Нажимает на кнопку добавления в корзину |
| 2 | Система | Меняет цвет кнопки, данные товара запоминаются и добавляются в корзину |
| 3 | Пользователь | Нажимает на кнопку получения информации |
| 4 | Система | Выводит диалоговое окно с информацией о способах заказа товара |
| 5 | Пользователь | Нажимает на иконку авторизации |
| 6 | Система | Выводит страницу входа |
| 7 | Пользователь | Нажимает на иконку корзины |
| 8 | Система | Выводит диалоговое окно корзины |
| 9 | Пользователь | Проверяет данные корзины, нажимает кнопку оформления заказа |
|  |  |  |
| Шаг | Действующее лицо | Действие |
| 10 | Система | Открывает диалоговое окно оформления заказа |
| 11 | Пользователей | Вводит информацию о заказе |
| 12 | Система | Сравнивает введенные данные с данными из базы данных, выводит сообщение о результате, отправляет письмо на почту пользователю с деталями заказа. |

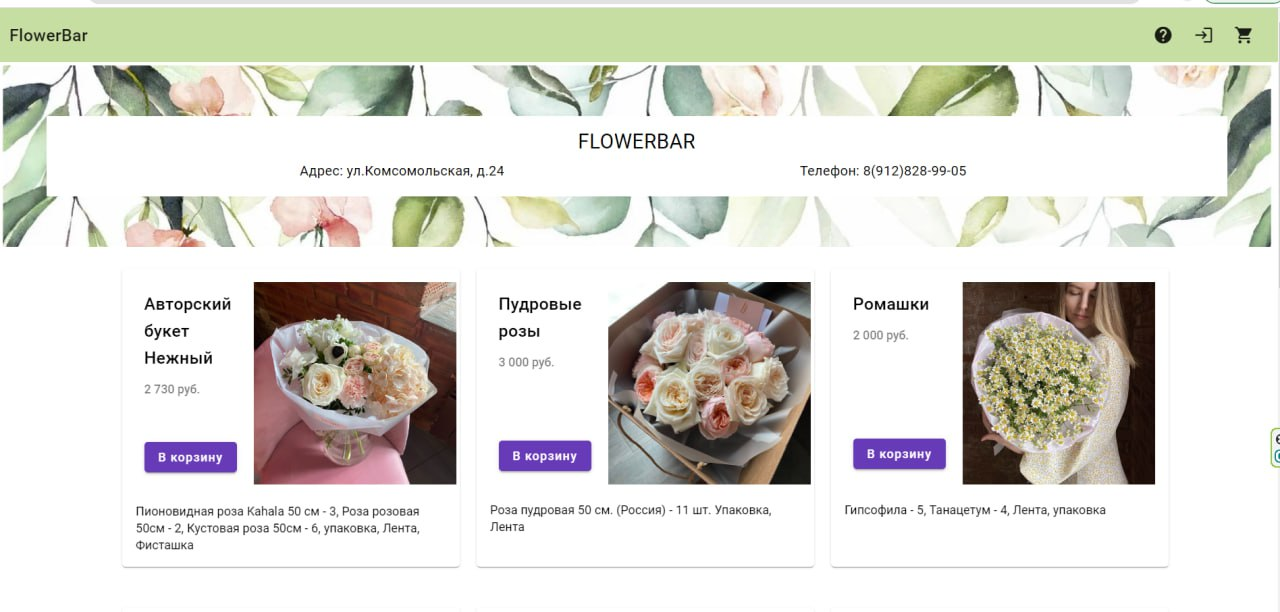
****Рисунок 28 – Главная страница

Таблица 3 - Этапы работы с главной страницей(продолжение)

Этапы работы со страницей менеджера прописаны в таблице 4. Страница регистрации представлена на рисунке 29.

Таблица 4 - Этапы работы со страницей менеджера

|  |  |  |
| --- | --- | --- |
| Номер шага | Действующее лицо | Действие |
| 1 | Пользователь | Просматривает список заказов. Нажимает на вкладку изменения статуса заказа |
| 2 | Система | Раскрывает форму изменения заказа |
| 3 | Пользователь | Вводит номер заказа и нажимает на кнопку изменения |
| Номер шага | Действующее лицо | Действие |
| 4 | Пользователь | Обновляет страницу |
| 5 | Система | Выводит измененные данные |
| 6 | Пользователь | Нажимает на вкладку удаления заказа |
| 7 | Система | Раскрывает форму изменения заказа |
| 8 | Пользователь | Вводит номер заказа и нажимает на кнопку удаления |
| 9 | Система | Сравнивает введенные данные с данными из базы данных и обновляет их |
| 10 | Система | Сравнивает введенные данные с данными из базы данных и удаляет их |
| 11 | Пользователь | Нажимает на иконку магазина |
| 12 | Система | Выводит главную страницу |
| 13 | Пользователь | Нажимает на кнопку списка пользователей |
| 14 | Система | Выводит диалоговое окно со списком пользователей |
| 15 | Пользователь | Нажимает на кнопку создания заявки на добавления товара для программиста |
| 16 | Система | Выводит диалоговое окно |
| 17 | Пользователь | Вводит необходимые данные и нажимает на кнопку добавления |
| 18 | Система | Сообщает о результате добавления. Заносит данные в базу данных. Отправляет на рабочую почту данные о добавлении товара |

Таблица 4 - Этапы работы со страницей менеджера(продолжение)

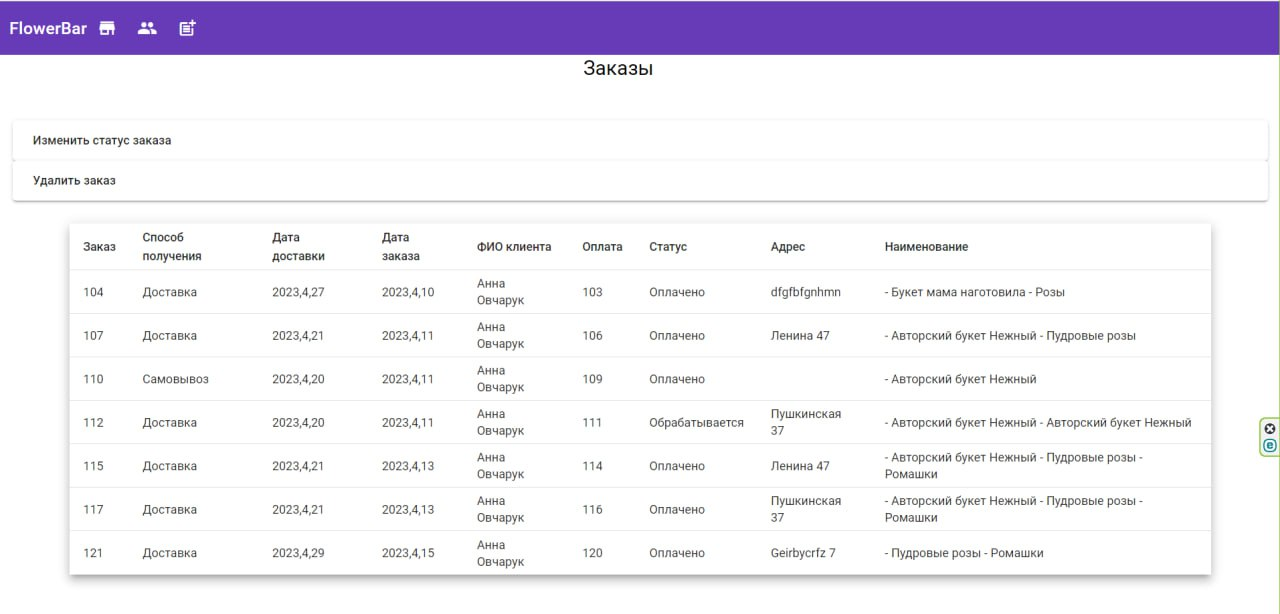


Рисунок 28 – Страница менеджера

Таблица 4 - Этапы работы со страницей менеджера(продолжение)

# Заключение

В ходе выполнения курсовой работы поставленные задачи были достигнуты. После этапа прототипирования, анализа, было разработано клиент-серверное веб-приложения онлайн покупки цветов с использованием фреймворка Angular для разработки клиентского веб-интерфейса и языка программирования Java и СУБД Postgresql для серверной части соответсвенно.

В первую очередь был создан каталог товаров, где пользователь может ознакомиться с информацией о цветах, а также, при нажатии на кнопку, добавить их в свою корзину. Выбранные позиции всегда отображаются в корзине с некоторой краткой. Данные функции доступны и для неавторизованного пользователя. Клиент может зарегистрироваться и авторизоваться. С серверной части приложение сохраняет информацию о пользователе.

Оформление заказа происходит после подтверждения пользователем состава его корзины. На форме заказа клиенту необходимо ввести данные о себе и данные для самого заказа. Он может выбрать дату заказа и способ доставки. Сразу же после оформления клиент получает результат в всплывающем окне, а также письмо с информацией о заказе на свою почту, указанную при оформлении заказа. Со стороны серверной части заказ сохраняется в базе данных и отображается на странице менеджера.

На странице менеджера реализован список заказов с полной информацией. Менеджер может изменить статус заказа или удалить его. У менеджера есть возможность просмотреть информацию о зарегистрированных пользователях. Реализован доступ к созданию заявки на добавление нового товара со стороны менеджера действующему программисту предприятия. После успешного создания заявки приходит письмо на рабочую почту магазина с информацией о товаре.

Приложение А

(обязательное)

Листинг программного кода реализации продуктов DTO

package com.example.dao;

import com.example.entity.Good;

import com.example.model.CartReciveModel;

import com.example.model.GodModel;

import com.example.repository.GodRepository;

import jakarta.inject.Inject;

import jakarta.inject.Singleton;

import java.util.ArrayList;

import java.util.List;

//

@Singleton

public class GodDTO {

@Inject

GodRepository godRepository;

public List<Good> GetAllGodd(){

return godRepository.findAll();

}

public List<GodModel> GetAllGod(){

List<Good> GodList = godRepository.findAll();

List<GodModel> GodModelList = new ArrayList<>();

for( Good god: GodList){

GodModelList.add(new GodModel(god.getId(),

god.getDescription(),

god.getPrice(),

god.getName()

));

}

return GodModelList;

}

public Good CreatNewGood(GodModel godModel){

Good good=new Good();

good.setDescription(godModel.getDescription());

good.setPrice(godModel.getPrice());

good.setName(godModel.getName());

godRepository.save(good);

return good;

}

public List<Good> GetAllGodToCart(List<CartReciveModel> cartReciveModels){

List<Good> goods = new ArrayList<>();

for (CartReciveModel cartReciveModel:cartReciveModels){

Good good = new Good();

good = godRepository.findByDescription(cartReciveModel.getName());

}

return goods;

}

}

Приложение Б

(обязательное)

Листинг кода маршрута отправки чека о покупке

package com.example.service;

import jakarta.inject.Singleton;

import javax.activation.DataHandler;

import javax.activation.DataSource;

import javax.activation.FileDataSource;

import javax.mail.\*;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;

import java.util.Properties;

@Singleton

public class SendFileService {

public int SendEmail(String Email,int sum,String FIO, int id) {

final String from = "flowerbarshop@mail.ru";

final String password = "A6fqJggGA2PHSvXhi5h5";//K1lnk2tJNhQDAa4TfsqX

String host = "smtp.mail.ru";

Properties props = new Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", host);

props.put("mail.smtp.port", "465");

props.put("mail.smtps.ssl.checkserveridentity", true);

props.put("mail.smtps.ssl.trust", "\*");

props.put("mail.smtp.ssl.enable", "true");

Session session = Session.getInstance(props,

new javax.mail.Authenticator(){

protected PasswordAuthentication getPassAuth(){

return new PasswordAuthentication(from, password);

}

});

try {

// Создание объекта MimeMessage по умолчанию

MimeMessage message = new MimeMessage(session);

// Установить От: поле заголовка

message.setFrom(new InternetAddress(from));

// Установить Кому: поле заголовка

message.addRecipient(Message.RecipientType.TO,new InternetAddress(Email));

// Установить тему: поле заголовка

message.setSubject("Чек ");

// Создание части сообщения

BodyPart messageBodyPart = new MimeBodyPart();

// Заполнение сообщения

messageBodyPart.setText("Уважаемый(aя), "+FIO+", чек оплаты заказа в цветочном магазине FlowerBar. Cумма заказа(номер- "+id+" ) составила: "+sum+"р. Благодарим вас за заказ.C вами свяжется наш флорист, сообщит реквизиты для оплаты. C Уважением FlowerBar, т. 8(912)828-93-45. Сообщение создано автоматически, отвечать не нужно.");

// Создание составного сообщения

Multipart multipart = new MimeMultipart();

// Установить часть текстового сообщения

multipart.addBodyPart(messageBodyPart);

// Отправить полные части сообщения

message.setContent(multipart );

// Отправить сообщение

try{

Transport.send(message,from,password);}catch(javax.mail.SendFailedException sendFailedException){return 2;}

System.out.println("Сообщение успешно отправлено");

} catch (MessagingException mex) {

mex.printStackTrace();

}

return 1;

}

}

Приложение Д

(справочное)

Библиографический список

1. Статьи Java: JavaRush [электронный ресурс] // URL: https://javarush.com/(дата обращения 11.04.2023).

2. Документация Ангуляр: angular.io [электронный ресурс] // URL: https://angular.io/guide/architecture-next-steps (дата обращения 11.04.2023).

3. Документация PostgreSQL: Компания Postgres Professional [электронный ресурс] // Последнее обновление 19.10.2021 URL: <https://postgrespro.ru/docs> (дата обращения 11.04.2023).