МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

WEB-приложение «Интернет-магазин брендовой одежды»

Выполнил студент Септилко Анастасия Антоновна

(Ф.И.О.)

Руководитель проекта преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[Введение 3](#_Toc40990089)

[1. Постановка задачи 4](#_Toc40990090)

[1.1. Алгоритмы решения 4](#_Toc40990091)

[1.2. Обзор прототипа 5](#_Toc40990092)

[2. Разработка архитектуры проекта 7](#_Toc40990093)

[2.1. Обобщенная структура 7](#_Toc40990094)

[2.2. Взаимосвязь всех компонентов 8](#_Toc40990095)

[2.3. Диаграмма UML 10](#_Toc40990096)

[3. Разработка функциональной модели и модели данных ПС 11](#_Toc40990097)

[3.1. Модель базы 11](#_Toc40990098)

[3.2. Выполняемые функции 11](#_Toc40990099)

[3.3. Регистрация и авторизация 15](#_Toc40990100)

[4. Руководство пользователя 16](#_Toc40990101)

[4.1. Регистрация пользователя 16](#_Toc40990102)

[4.2. Аутентификация и авторизация пользователя 17](#_Toc40990103)

[4.3. Оформление заказа 18](#_Toc40990104)

[4.4. Онлайн-консультант 20](#_Toc40990105)

[5. Тестирование 22](#_Toc40990106)

[Заключение 24](#_Toc40990107)

[Список используемых источников 25](#_Toc40990108)

[Приложение А 26](#_Toc40990109)

[Приложение Б 27](#_Toc40990110)

[Приложение В 28](#_Toc40990111)

# **Введение**

В данном курсовом проекте разработано WEB-приложение «Интернет-магазин брендовой одежды». Логически оно разделено на две части: серверную, написанную на Node.js, и клиентскую, в которой использовался язык JavaScript с внедрением  шаблонизатора [Handlebars](http://handlebarsjs.com/), физически же WEB-приложение представляет собой одно приложение.

В современном мире сложно представить человека, который не пользуется интернетом. Люди привыкли все делать через интернет, например, оплачивать коммунальные услуги, класть деньги на интернет, телефон, покупать продукты, осуществлять доставки, даже одеваться сейчас можно, используя интернет-магазины, где ты сможешь выбрать понравившийся товар и заказать его, не выходя из дома сидя на диване.

Целью курсового проекта является разработка WEB-приложения, которое поможет вам осуществить заказ. Я сделала выбор своей курсовой темы в связи с актуальностью данной задачи в современном мире. Данное приложение содержит тот функционал, который поможет вам осуществлять покупки, не выходя из дома, тем самым экономить ваше время.

Пояснительная записка курсового проекта состоит из 28 страниц, 22 рисунков, 3 приложений, 9 источников литературы.

Основная цель курсового проекта: разработка web-приложения «Интернет-магазин».

В первом разделе рассматриваются основные технологии, которые использовались в разработке данного приложения, а также его прототипы и актуальность задачи.

Во втором разделе описана архитектура курсового проекта.

В третьем разделе предоставлена информация о разработанных объектах базы данных.

Четвертый раздел содержит руководство пользователя для разработанного клиентского приложения.

В пятом разделе представлены результаты тестирования приложения.

В заключении описывается результат курсового проектирования и задачи, которые были решены в ходе разработки приложения.

1. **Постановка задачи**

Главная задача курсового проекта является разработка WEB- приложения, позволяющего пользователю просматривать каталог товаров, который разделен на категории для мужчин, для женщин и для детей, добавлять товары в избранное и в корзину, а также удалять из избранного и из корзины, оформлять заказ. В данном курсовом проекте требовалось реализовать следующие задачи:

* сохранение рабочей информации в централизованной базе данных;
* организация регистрации и входа в систему со стороны пользователей;
* реализация архитектуры MVC;
* возможность просмотра, добавления, удаления, редактирования товарных позиций;
* осуществление заказа.

## **Алгоритмы решения**

При реализации курсового проекта использовались технологии Node.js[5], фреймворк Express[6], ORM библиотека sequelize[2], WebSoket[1] для реализации back-end, для front-end части Bootstrap [7], JQuery[8], JavaScript[3], [Handlebars](http://handlebarsjs.com/)[4], CSS. MS SQL в качестве базы данных.

Node.js –  программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (транслирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) в [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) взаимодействовать с устройствами [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API) (написанный на [C++](https://ru.wikipedia.org/wiki/C%2B%2B)), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль [веб-сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит [событийно-ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F).

Express – [фреймворк web-приложений](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для Node.js, реализованный как [свободное и открытое программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%B8_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) под [лицензией MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT). Он спроектирован для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [API](https://ru.wikipedia.org/wiki/API). Де-факто является стандартным каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описывает его как созданный на основе написанного на языке [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby) каркаса [Sinatra](https://ru.wikipedia.org/wiki/Sinatra" \o "Sinatra), подразумевая, что он минималистичен и включает большое число подключаемых [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD).

Sequelize– это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в базе данных и отношений между ними с классами. При использовании Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами.

WebSoket – это передовая технология, которая позволяет создавать интерактивное соединение между клиентом (браузером) и сервером для обмена сообщениями в режиме реального времени. Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно уникальной.

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений, который включает в себя HTML и CSS – шаблоны оформления для веб-форм, кнопок, меток, слайд шоу, блоков навигации и прочих компонентов веб-интерфейса.

JQuery– [набор функций JavaScript](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующийся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX)

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

[Handlebars](http://handlebarsjs.com/) – это шаблонный процессор, который динамически генерирует вашу HTML-страницу, что экономит ваше время на ручном обновлении.

## **Обзор прототипа**

В наше время в сети Интернет можно найти любую информацию, поэтому было решено поискать аналоги проектируемого приложения в иных ведущих интернет-магазинах страны.

В качестве прототипа был выбран интернет-магазин «Lamoda», который представлен на рисунке 1.1. Это приложение предназначено для осуществления заказа, покупки товаров и доставки. Достаточно удобная система покупки и доставки товаров. Приложение имеет довольно красивый и понятный любому пользователю интерфейс и функционал. Как пример был взят для реализации онлайн-консультант магазина.

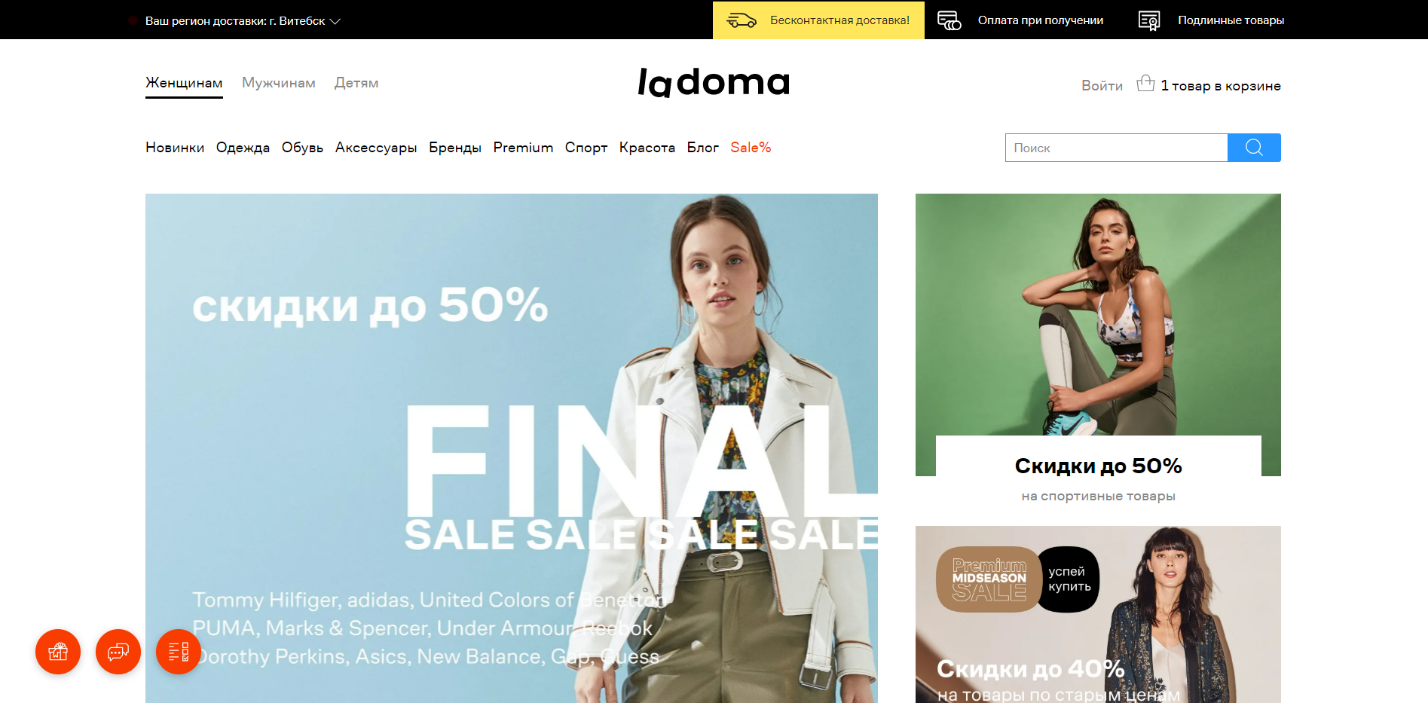


Рисунок 1.1 – Интернет-магазин «Lamoda»

Еще один интернет-магазин помог мне определиться с функциональностью моего WEB-приложения – это магазин «Wildberries» (рисунок 1.2).

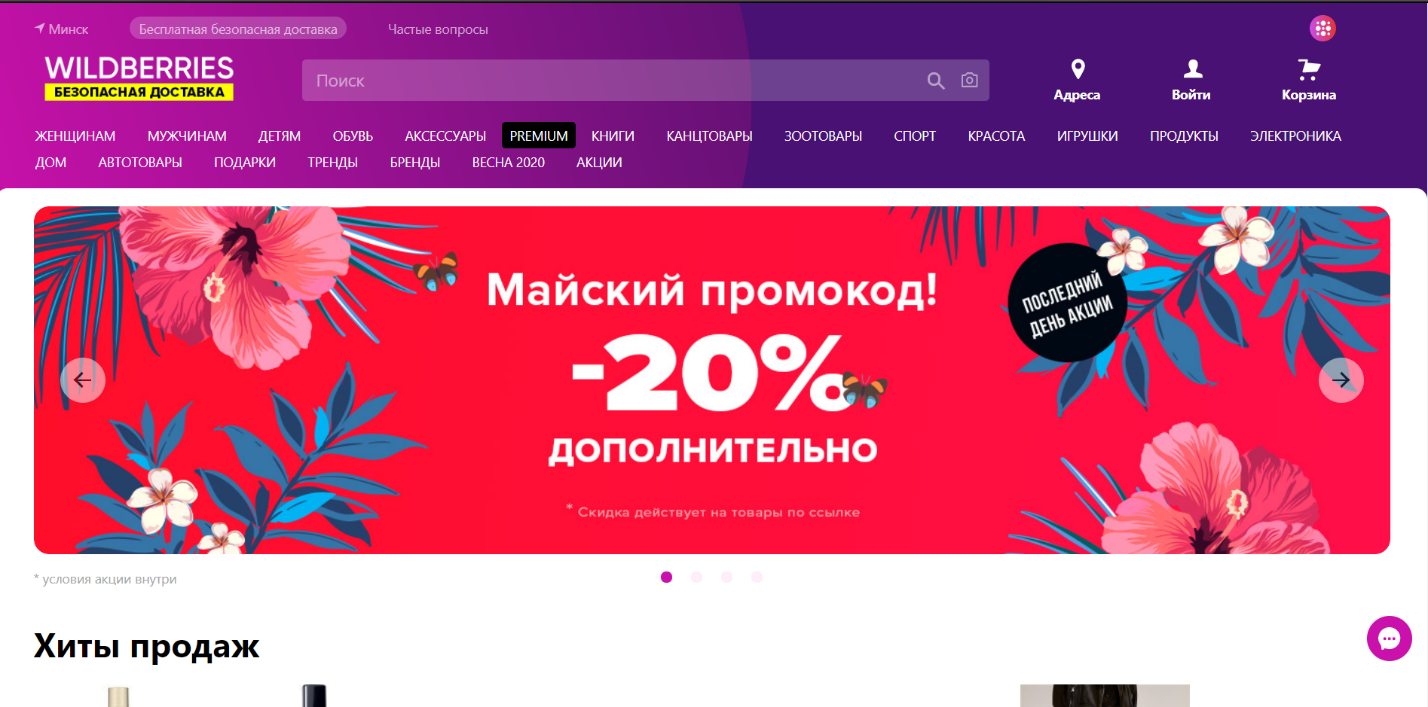


Рисунок 1.2 – Интернет-магазин «Wildberries»

# **Разработка архитектуры проекта**

Разработка архитектуры проекта – одна из важнейших задач в процессе работы над приложением, так как в зависимости от неё определяется уровень зависимости компонентами приложения, и насколько легко расширяемы её составные части.

Архитектура проекта – это его строение как оно видно (или должно быть видно) из вне его, т.е. представление программного средства как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы. Разработка архитектуры является первым этапом борьбы со сложностью программного средства, на котором реализуется принцип выделения относительно независимых компонент.

Основные задачи разработки архитектуры проекта:

* Выделение программных подсистем и отображение на них внешних функций (заданных по внешнем описании) программного средства;
* Определение способов взаимодействия между выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

## **Обобщенная структура**

Курсовой проект построен на многоуровневой архитектуре (N-layer architecture)[9]. Данная архитектура основана на следующих принципах:

* Проектирование чётко устанавливает разграничение функций между уровнями;
* Нижние уровни независимы от верхних уровней;
* Верхние уровни вызывают функции нижних уровней, но при этом взаимодействуют только соседние уровни иерархии.

Для данного курсового проекта были разработаны следующие уровни:

* Уровень «Модель» – описание таблиц в базе данных;
* Уровень «Представление» – отвечает за отображение данных модели пользователю, реагируя на изменения модели;
* Уровень «Контроллер» – описание классов, методы которых предназначены для создания ответа пользователю по его запросу.
* Уровень «Авторизация» – назначение пользователю прав и привилегий для работы с объектами базы данных.
* Уровень «Аутентификация» – проверка подлинности пользователя.

Каждый запрос клиента проходит как минимум через 3 уровня («Аутентификация», «Авторизация» и «Контроллер»), после чего клиент получает ответ. В случае успешной аутентификации и авторизации клиента сервер также может задействовать другие уровни приложения.

В приложении используется паттерн MVC:

Паттерн MVC является одним из распространенных паттернов, применяемых в веб-приложениях. В том числе он применяется и в приложениях на Node.js.

Паттерн MVC включает ряд компонентов:

* Модели определяют структуру и логику используемых данных
* Представления (views) определяет визуальную часть, как данные будут отображаться
* Контроллеры обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляет в ответ клиенту некоторый результат обработки, нередко в виде html-кода.
* Система маршрутизация как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

В общем случае, когда к приложению приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращаться к данным через модели и для рендеринга ответа использовать представления. Результат обработки контроллера отправляется в ответ клиенту. Нередко ответ представляет html-страницу, которую пользователь видит в своем браузере.

## **Взаимосвязь всех компонентов**

В решении курсового проекта была использована n-layer архитектура. Представлена на рисунке 2.1. Эта архитектура выбрана, так как она идеально подходит для приложений с длительным жизненным циклом и сложной бизнес логикой. Для реализации серверной части была использована N-Layer архитектура. Ее схематическое представление продемонстрировано на рисунке 2.1. Данный курсовой проект построен по примеру ниже приведенной схемы классической трехуровневой системы.

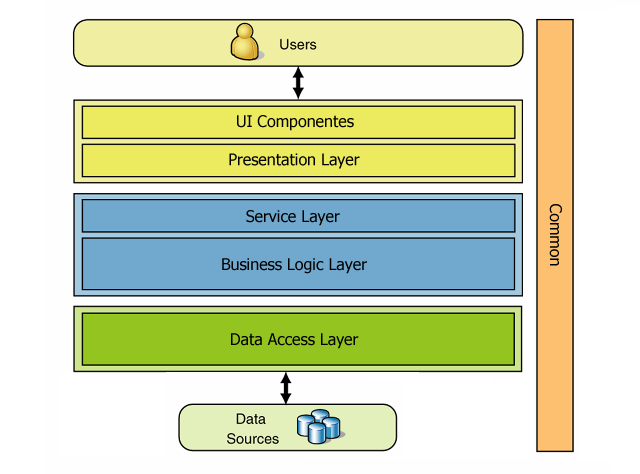


Рисунок 2.1 – Общая схема N-Layer архитектуры

Data Access Layer (уровень доступа к данным) содержит сущности, соответствующие таблицам базы данных, а также схемы взаимодействия друг с другом.

Business Logic Layer (уровень бизнес-логики) при тесном взаимодействии с Service Layer (уровень сервисов) являются посредниками между сущностями базы данных и моделями для UI, сочетая в себе сервисы и классы, реализующий основной функционал всего разрабатываемого приложения.

Presentation Layer (уровень представления), поддерживаемый UI Components (User Interface Components – компоненты пользовательского интерфейса), представляет собой генерирующуюся на сервере разметку или получаемые с уровня бизнес-логики данные для REST API в формате JSON, преобразующиеся впоследствии в разметку на стороне клиента. Уровень представления состоит состоящее из контроллеров, координирующих пользовательские запросы, и представлений, формирующееся в разметку HTML**.**

## **Диаграмма UML**

ДиаграммаUML–это графическое представление набора элементов,изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграмма использования представлена на рисунке 2.2.

В языке UML вариант использования изображается в виде овала, помеченного именем представляемого варианта. Варианты использования могут быть связаны с участвующими в них действующими лицами (actors), изображаемыми в виде человечков и представляющими различные роли пользователей системы или внешние системы, взаимодействующие с ней.

Варианты использования могут быть связаны друг с другом тремя видами связей: обобщением (generalization), расширением (extend relationship) и включением (include relationship). Действующие лица также могут быть связаны друг с другом с помощью связей обобщения (generalization).

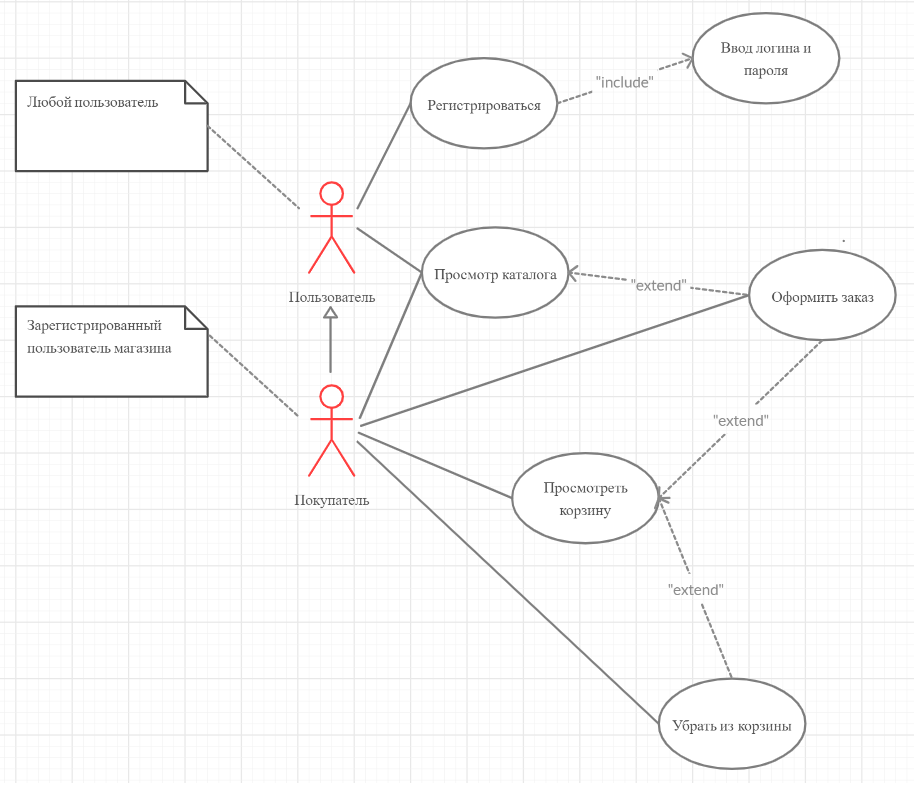


Рисунок 2.2 – Диаграмма использования

1. **Разработка** **функциональной модели и модели данных ПС**

Для разработки и управления базой данных курсового проекта использовалась объектно-реляционная система управления базами данных «MS SQL Server».

Выбор данного системного продукта произошел по нескольким очевидным причинам: легкая инсталляция, бесплатна для разработки, раннее полученные знания по использованию данной базы данных.

База данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Основные функции СУБД:

* определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки;
* предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация);
* обеспечение логической и физической независимости данных;
* защита логической целостности базы данных;
* защита физической целостности;
* управление полномочиями пользователей на доступ к базе данных;
* синхронизация работы нескольких пользователей;
* управление ресурсами среды хранения;
* поддержка деятельности системного персонала.

Примеры СУБД: Oracle, MS SQL Server, Microsoft Access, MySql, PostgreSQL, MongoDB и так далее.

## **Модель базы**

Для базы данных QDEW\_DB было разработано 6 таблиц. На рисунке 3.1 предоставлена структура связей внешних ключей и первичных ключей. В приложении А представлена полная диаграмма таблиц.

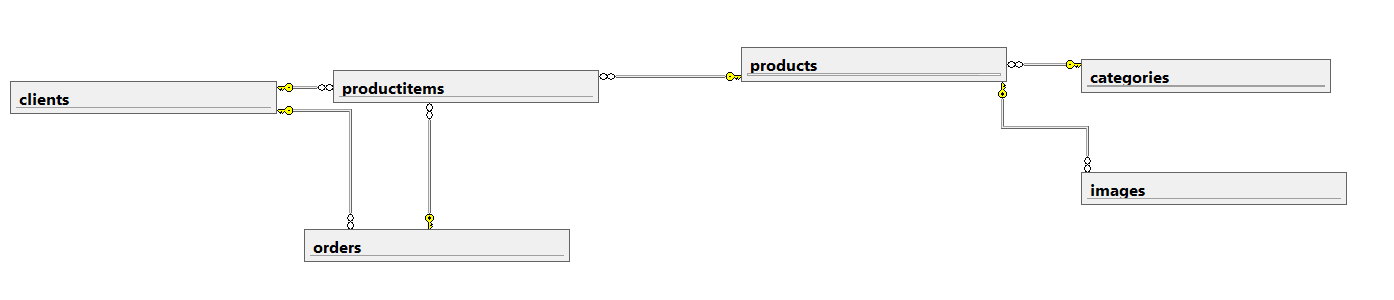


Рисунок 3.1 – Структура базы данных

## **Выполняемые функции**

На рисунке 3.2 представлена структура таблицы Clients, которая содержит информацию о пользователях системы.

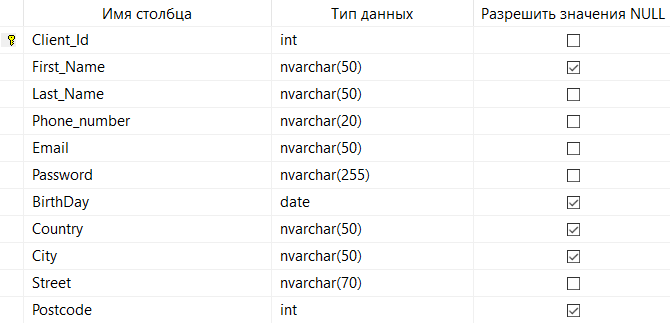


Рисунок 3.2 – Таблица Clients

Таблица включает поля:

* Client\_Id – первичный ключ, идентификатор пользователя;
* First\_Name – имя пользователя в системе;
* Last\_Name – фамилия пользователя;
* Phone\_number – мобильный телефон пользователя;
* Email – почта пользователя;
* Password – пароль пользователя;
* BirthDay – дата дня рождения пользователя;
* Country – страна в которой проживает пользователь;
* City – город в котором проживает пользователь;
* Street – улица на которой проживает пользователь;
* Postcode – почтовый код.

На рисунке 3.3 представлена структура таблицы Orders, которая содержит информацию для осуществления заказа.

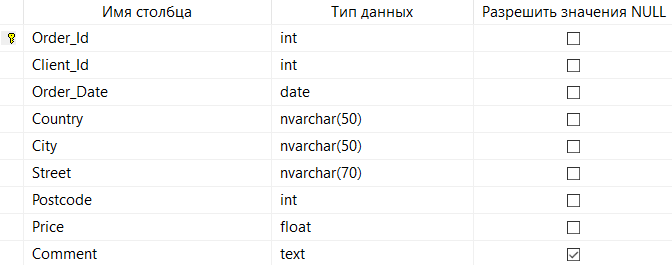


Рисунок 3.3 – Таблица Orders

Таблица включает поля:

* Order\_Id – первичный ключ, идентификатор заказа;
* Client\_Id – вторичный ключ, идентификатор пользователя;
* Order\_Date – дата оформления заказа;
* Country – страна куда оформлять доставку;
* City – город куда оформлять доставку;
* Street – улица куда оформлять доставку;
* Postcode – почтовый код;
* Price – общая цена заказа;
* Comment – комментарий к заказу (возможно дополнительные пожелание пользователя).

На рисунке 3.4 представлена структура таблицы ProductItems, которая содержит информацию о продукте, а именно добавлен ли продукт в корзину, добавлен ли продукт в избранное.

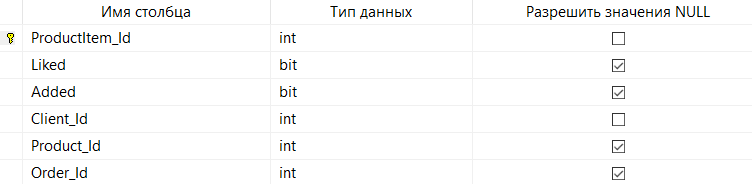


Рисунок 3.4 – Таблица ProductItems

Таблица включает поля:

* ProductItem\_Id – первичный ключ, идентификатор таблицы ProductItem;
* Liked – если 1, то продукт добавлен в избранное, если 0, то не добавлен в избранное;
* Added – если 1, то продукт добавлен в корзину, если 0, то не добавлен в корзину;
* Client\_Id – вторичный ключ, идентификатор пользователя;
* Product\_Id – вторичный ключ, идентификатор продукта;
* Order\_Id – вторичный ключ, идентификатор заказа.

На рисунке 3.5 представлена структура таблицы Products, которая содержит информацию о товарах.

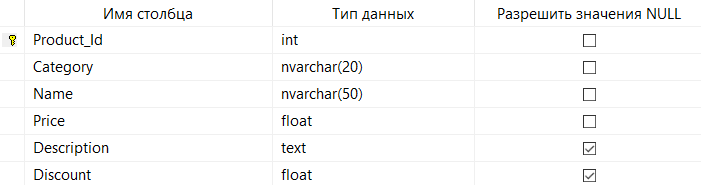


Рисунок 3.5 – Таблица Products

Таблица включает поля:

* Product\_Id – первичный ключ, идентификатор товара;
* Category – вторичный ключ, хранит в себе категории товаров;
* Name – название товара;
* Price – цена товара;
* Description – описание товара;
* Discount – скидка товара (если есть).

На рисунке 3.6 представлена структура таблицы Categories, которая содержит категории товаров.



Рисунок 3.6 – Таблица Categories

Таблица включает поле:

* Name – первичный ключ, название категории.

На рисунке 3.7 представлена структура таблицы Image, которая содержит изображение товаров.

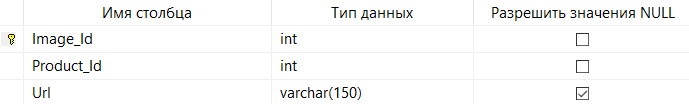


Рисунок 3.7 – Таблица Image

Таблица включает поля:

* Image\_Id – первичный ключ, идентификатор изображения;
* Product\_Id – вторичный ключ, идентификатор товара;
* Url – хранит в себе путь изображения.

## **Регистрация и авторизация**

Регистрация пользователя происходит в разделе Registration, листинг которого находится в приложении Б. При регистрации предусмотрена валидация данных. Логин и пароль пользователя уникальны. Каждый пользователь сначала должен зарегистрироваться, после чего он сможет авторизоваться.

Авторизация пользователя происходит на странице Login, листинг приведен в приложение В. Также предусмотрена валидация данных.

1. **Руководство пользователя**

Данная глава содержит описание некоторых функций приложения для более легкого восприятия конечного пользователя.

* 1. **Регистрация пользователя**

Для регистрации необходимо заполнить форму, которая продемонстрирована на рисунке 4.1.

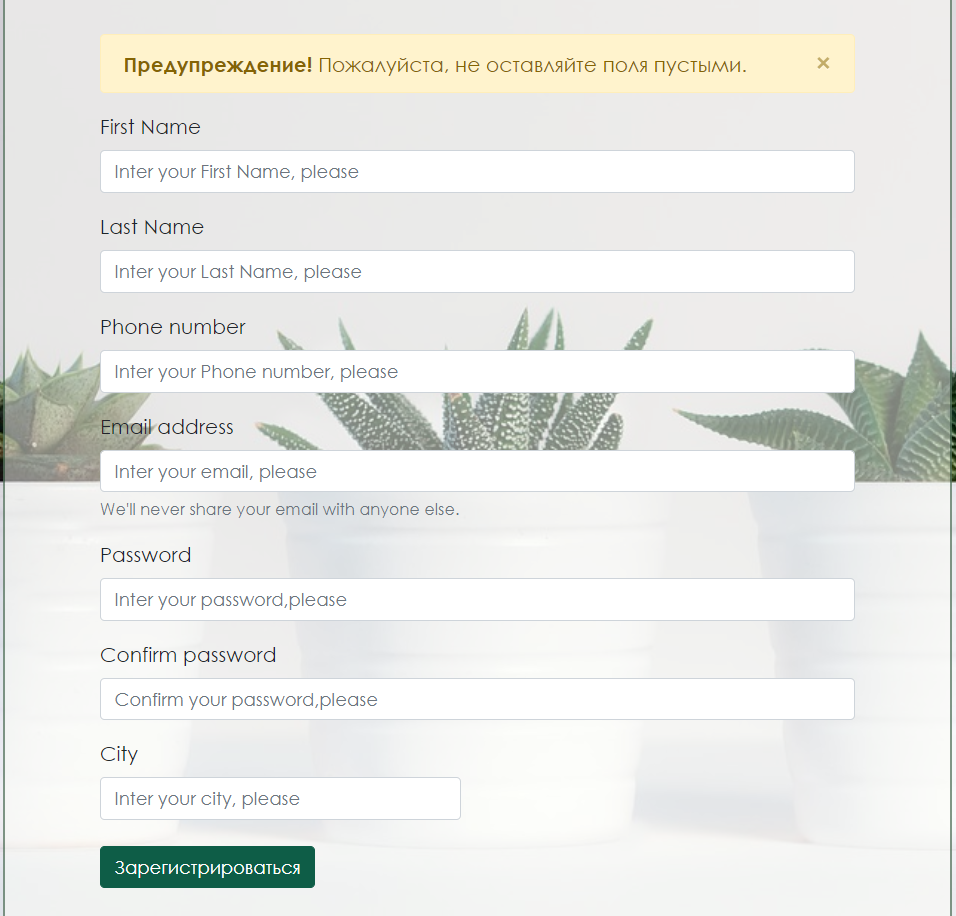


Рисунок 4.1 – Форма регистрации

После отправки пользователем запроса на регистрацию, происходит валидация всех полей формы. Если валидация будет неуспешной, пользователь увидит сообщения об ошибках в соответствующих полях для ввода информации. В случае успешной регистрации пользователь переадресовывается на страницу входа в приложение.

* 1. **Аутентификация и авторизация пользователя**

Для аутентификации необходимо ввести пароль, логин и нажать на кнопку «Войти», что позволит проверить, существует ли пользователь с введенными данными в системе. Все поля являются обязательными для заполнения.

Если аутентификация прошла успешно и соблюдены все выше описанные правила, то происходит перенаправление на главную страницу, что означает, что пользователь был успешно авторизован. Скриншот формы авторизации представлен на рисунке 4.2.

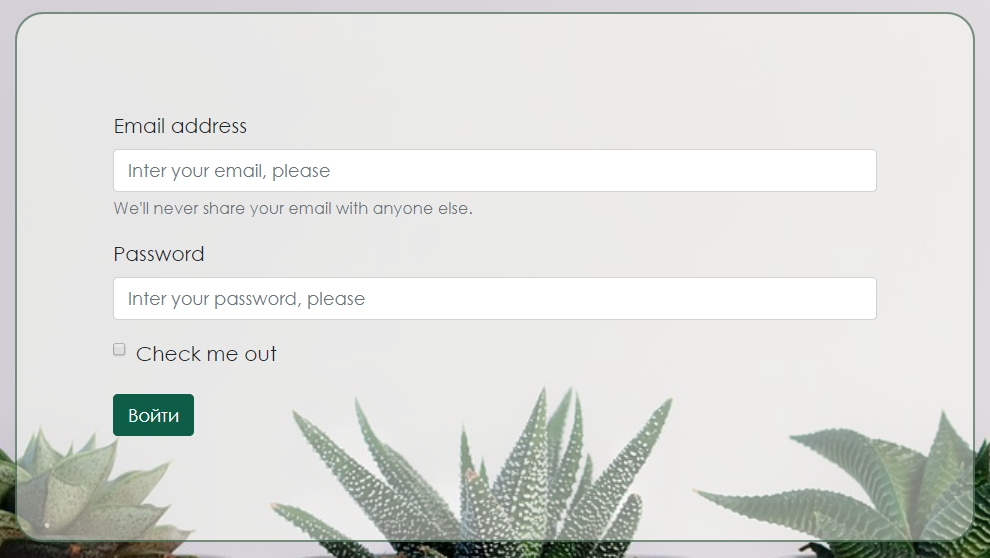


Рисунок 4.2 – Фрагмент экрана с формой для авторизации пользователя

При входе в систему авторизированному пользователю выдается токен, который записывается в куки, тем самым позволяя разграничить возможности пользователей на сайте.

В данном курсовом проекте логика приложения задумана так, что пользователь, который не вошел в систему имеет доступ только к просмотру товарных позиций. Если пользователь авторизировался, то у него появляется возможность добавления (удаления) товаров в избранные товары, добавления (удаления) товаров в корзину, возможность осуществлять заказ.

* 1. **Оформление заказа**

Для того, чтобы пользователь смог оформить заказ ему необходимо вначале войти в интернет-магазин под своим аккаунтом, потому что, если пользователь не авторизирован он лишается возможности добавить продукт в корзину, тем самым оформить заказ. После того как пользователь вошел в систему ему нужно выбрать понравившийся товар и добавить его в корзину, нажав на иконку «Корзина» (рисунок 4.3).

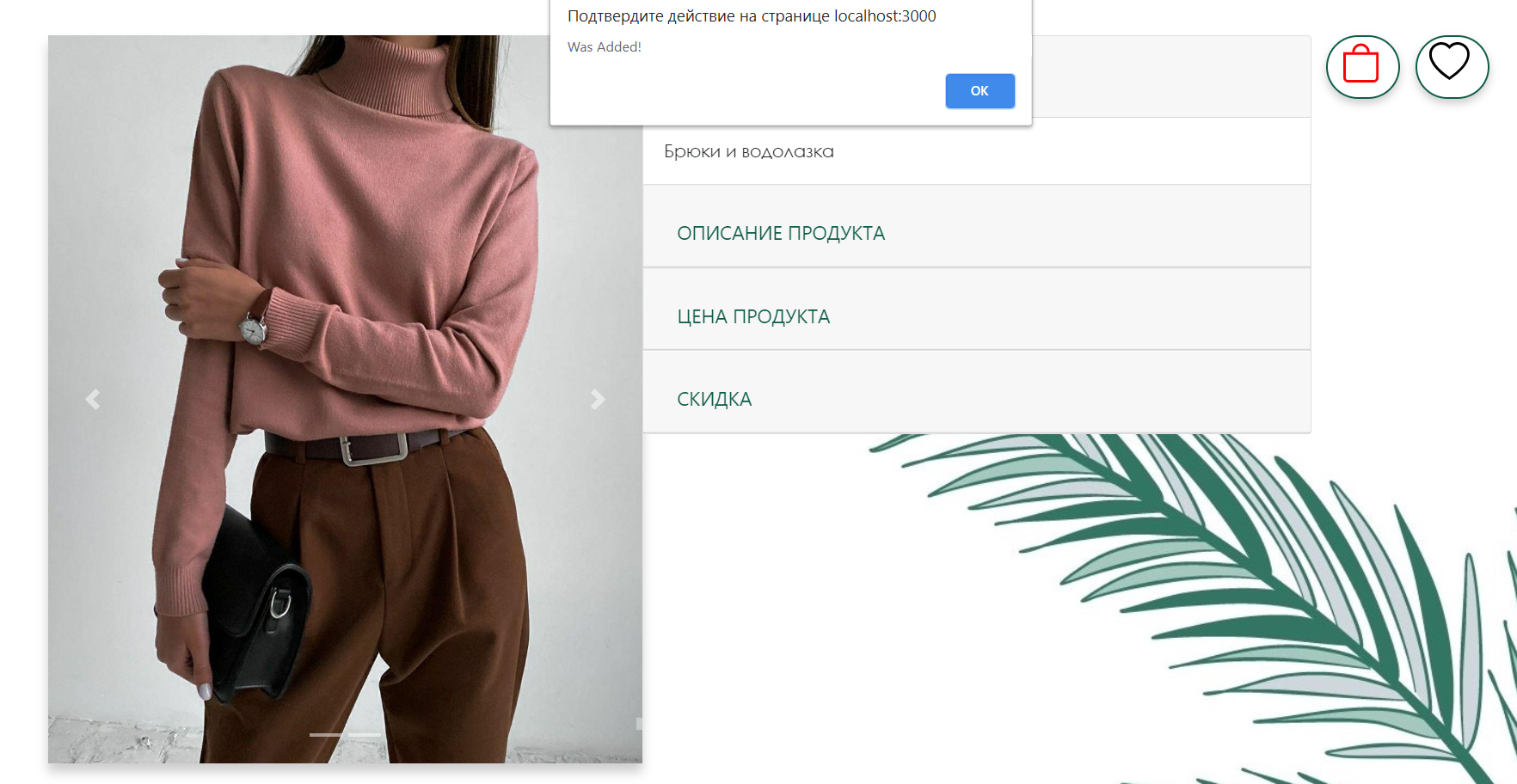


Рисунок 4.3 – Добавление товара в корзину

После того как пользователь добавил в корзину товары, которые хочет приобрести ему следует перейти в корзину. Корзина находится на главной странице справа в верхнем углу. Перейдя в корзину внизу нужно нажать на кнопку «Оформить заказ». После нажатия на данную кнопку появится форма для заполнения (рисунок 4.4).

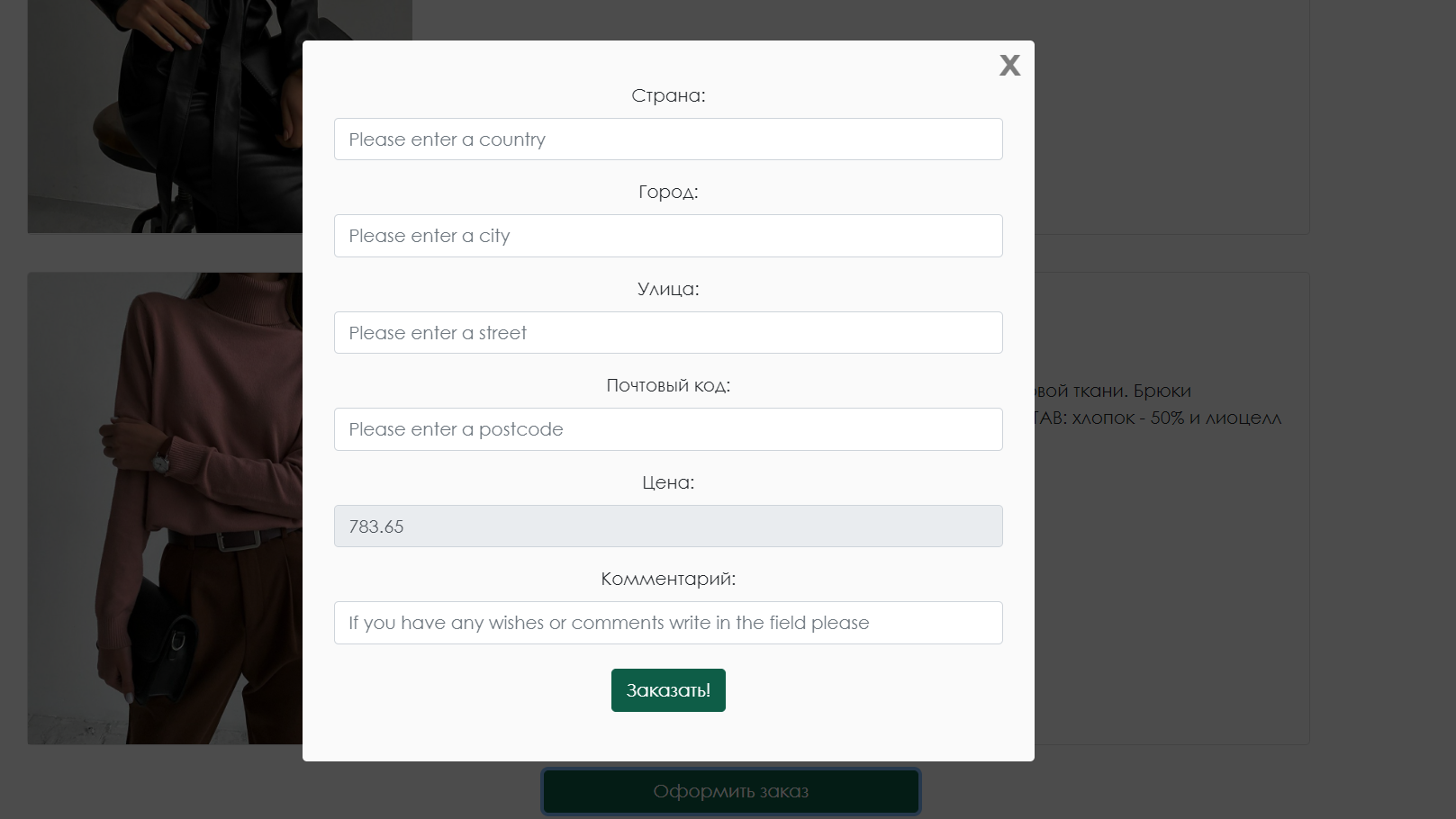


Рисунок 4.4 – Форма для оформления заказа

Можно заметить, что цену изменить невозможно, программа сама подсчитывает сумму всех товаров в корзине и выводит пользователю итоговую цену. При нажатии на поле «Цена» будет небольшое уведомление о том, что способ оплаты имеет два типа и оплата производиться с курьером. Уведомление можно увидеть на рисунке 4.5.



Рисунок 4.5 – Уведомление о способе оплаты

Когда все поля буду корректно заполнены пользователем, необходимо нажать на кнопку «Заказать». Ваш заказ будет оформлен, пользователя уведомят об этом сообщением, который продемонстрирован на рисунке 4.6.

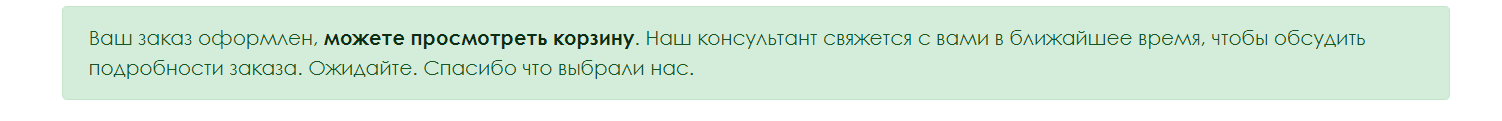


Рисунок 4.6 – Уведомление о том, что заказ успешно оформлен

## **Онлайн-консультант**

В данном курсовом проекте был разработан онлайн-консультант для пользователей, которые не знают, как пользоваться интернет-магазином. Консультант помогает пользователю оформить заказ.

Онлайн консультант был реализован с помощью WebSoket. Более подробно работа WebSoket описана в первой главе.

Чтобы воспользоваться помощью консультанта вам нужно его открыть, нажав на вторую иконку слева внизу главной страницы (рисунок 4.7).

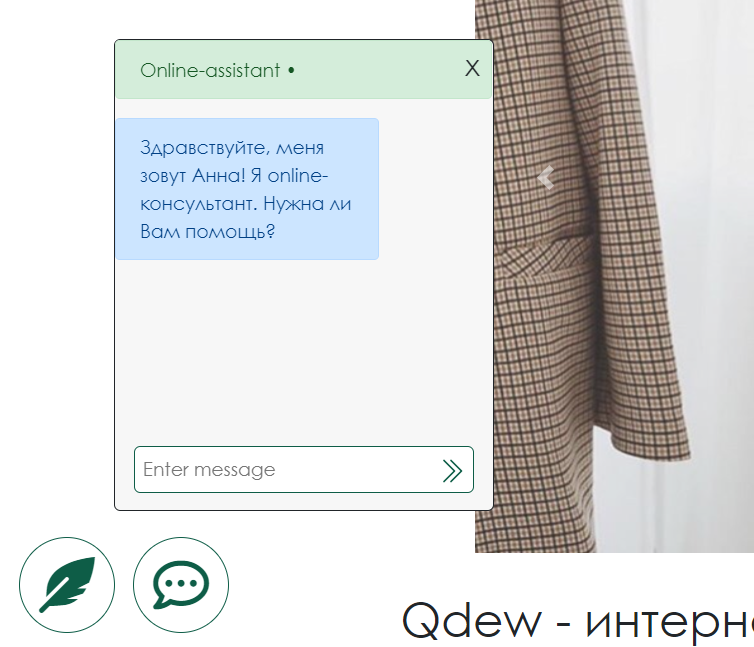


Рисунок 4.7 – Онлайн-консультант

После этого стоит просто написать ему «да / yes» или «нет / no». Если вы напишете ему «да», то консультант подробно вам опишет как пользоваться интернет-магазином и как осуществить заказ (рисунок 4.8).

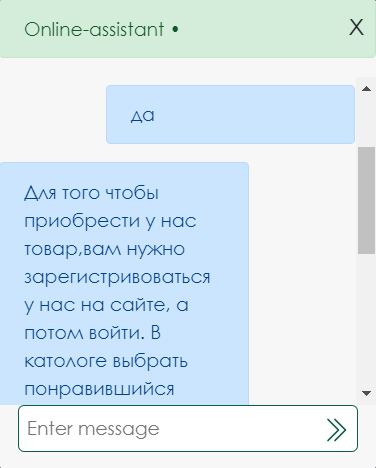


Рисунок 4.8 – Ответ онлайн-консультанта

# **Тестирование**

Для обеспечения корректности работы программы, обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

В курсовом проекте организована валидация полей на случай неочевидного или неправильного поведения пользователя.

Валидация приложения – это один из основных гарантов надёжности приложения и всякий программист должен предусмотреть и предотвратить непредвиденное поведение пользователя.

При регистрации обрабатываются вводимые данные пользователя. Так как поле «email» является уникальным, в базе данных не могут храниться пользователи с одинаковыми логинами, если в базе данных уже существует пользователь с таким же логином, то клиента уведомят об ошибке, это продемонстрировано на рисунке 5.1.

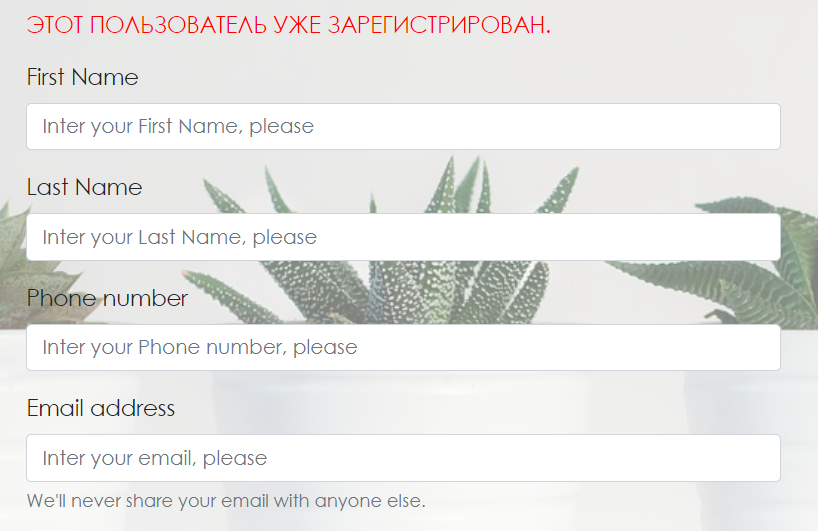


Рисунок 5.1 – Некорректный логин

Также возникают ошибки, если пользователь пытается ввести неверные данные в поле «Confirm password» (рисунок 5.2). Введенный пароль в поле «Password» должен полностью совпадать с введенным паролем в поле «Confirm password».

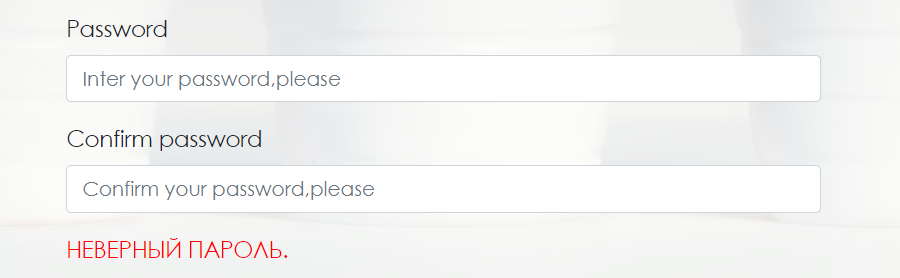


Рисунок 5.2 – Несовпадение паролей

При входе в систему также обрабатываются вводимые данные пользователя. Если пользователь неверно ввел данные в поля для входа, его уведомят об этом, пример уведомления об ошибке предоставлен на рисунке 5.3.

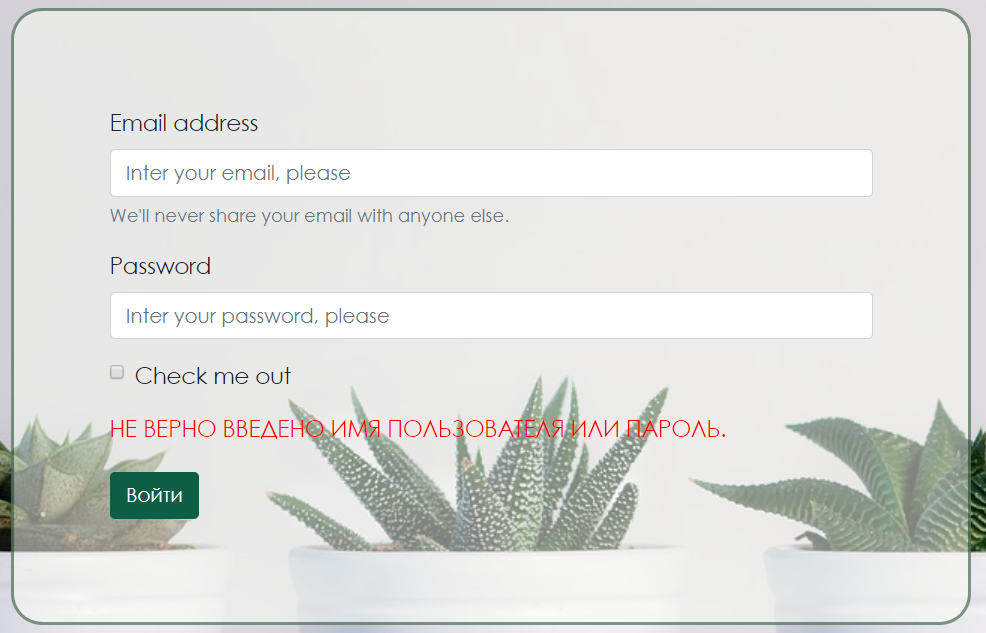


Рисунок 5.3 – Неверно введены данные

# **Заключение**

В данном курсовом проекте была разработана база данных и WEB-приложение «Интернет-магазин брендовой одежды», которое позволяет пользователю просматривать каталог товаров, который разделен на категории для мужчин, для женщин и для детей, добавлять товары в избранное и в корзину, а также удалять из избранного и из корзины, оформлять заказ. Были выполнены следующие задачи:

* сохранение рабочей информации в централизованной базе данных;
* организация регистрации и входа в систему со стороны пользователей;
* реализация архитектуры MVC;
* возможность просмотра, добавления, удаления, редактирования товарных позиций;
* осуществление заказа.

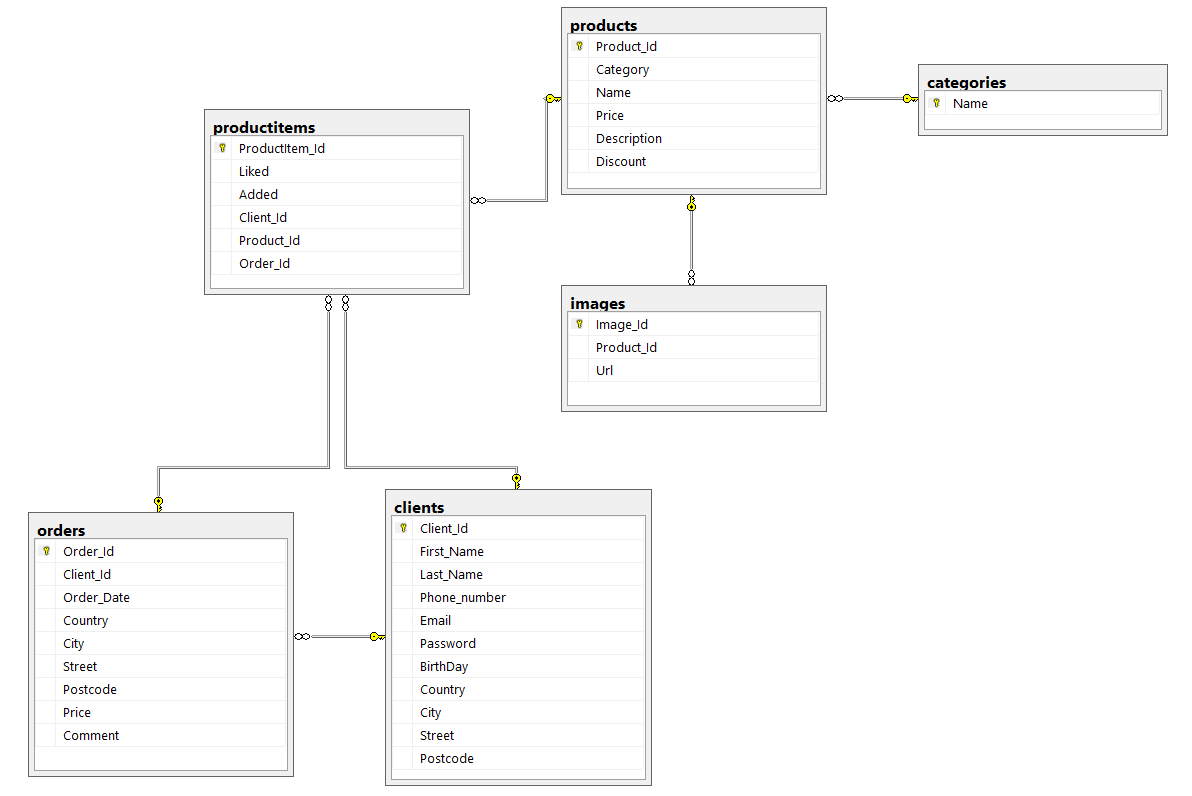
Данное программное средство использует технологии Node.js, фреймворк Express, ORM библиотека sequelize, WebSoket для реализации back-end, для front-end части Bootstrap, JQuery, JavaScript, [Handlebars](http://handlebarsjs.com/), CSS. MS SQL в качестве базы данных.

В соответствии с полученным результатом работы программы, можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список используемых источников**

1. Асинхронный веб, или что такое веб-сокеты [Электронный ресурс]. — Режим доступа: <https://tproger.ru/translations/what-are-web-sockets/> Дата доступа: 13.03.2020.
2. Что такое Sequelize [Электронный ресурс]. — Режим доступа: <https://metanit.com/web/nodejs/9.1.php> Дата доступа: 10.03.2020.
3. JavaScript [Электронный ресурс]. — Режим доступа:  <https://ru.wikipedia.org/wiki/JavaScript> Дата доступа: 02.03.2020.
4. Введение в Handlebars [Электронный ресурс]. — Режим доступа: <https://code.tutsplus.com/ru/tutorials/an-introduction-to-handlebars--net-27761> Дата доступа: 27.03.2020.
5. Что такое Node.js [Электронный ресурс]. — Режим доступа: <https://metanit.com/web/nodejs/1.1.php> Дата доступа: 07.03.2020.
6. Начало работы с Express [Электронный ресурс]. — Режим доступа: <https://metanit.com/web/nodejs/4.1.php> Дата доступа: 07.03.2020.
7. Bootstrap [Электронный ресурс]. — Режим доступа: <https://getbootstrap.com/> Дата доступа: 27.03.2020.
8. JQuery [Электронный ресурс]. — Режим доступа: <https://jqueryui.com/draggable/> Дата доступа: 07.04.2020.
9. N-layer архитектура [Электронный ресурс]. — Режим доступа: <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier> Дата доступа: 07.03.2020.

# **Приложение А**



# **Приложение Б**

register(request, response) {  
 if (!checkPasswordsMatch(request.body.password, request.body.confirm, response)) {  
 return;  
 }  
  
 db.Clients.*findOne*({  
 where: {  
 Email: request.body.username  
 }  
 }).then(client => {  
 if (client) {  
 response.render('register', {  
 title: 'Register',  
 layout: 'authorization',  
 errorIsUser: 'Этот пользователь уже зарегистрирован.'  
 });  
 } else {  
 db.Clients.create({  
 First\_Name: request.body.firstName,  
 Last\_Name: request.body.lastName,  
 Phone\_number: request.body.phoneNumber,  
 Email: request.body.username,  
 Password: request.body.password,  
 City: request.body.city  
 }).then(user => {  
 response.render('login', {  
 title: 'Login',  
 layout: 'authorization' });  
 });}  
 });  
}

function checkPasswordsMatch(password, confirm, response) {  
 if (password !== confirm) {  
 response.render('register', {  
 layout: 'authorization',  
 error: 'Неверный пароль.'  
 });  
 return false;  
 } else return true;  
}

# **Приложение В**

login(request, response) {  
  
 db.Clients.*findOne*({  
 where: {  
 Email: request.body.username,  
 Password: request.body.password  
 }  
 }).then(client => {  
 if (client) {  
 jwt.sign({ user: client }, jwtSecret, function(err, token) {  
 if (err) {  
 *console*.log('Generate token error');  
 } else {  
 response.cookie('token', token);  
 }  
 response.redirect('/');  
 });  
 } else {  
 response.render('login', {  
 title: 'Login',  
 layout: 'authorization',  
 error: 'Не верно введено имя пользователя или пароль.'  
 });  
 }  
 });  
}