МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра компьютерных технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине «СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

на тему «Разработка онлайн-магазина по продаже брендовой одежды»

Выполнил: студент

группы ЗКТ-45-21

Башкиров Денис Владимирович

Проверил: доцент кафедры компьютерных технологий, к.ф.-м.н., Тихонов Сергей Владимирович

Чебоксары, 2025

**Содержание**

[**Введение** 3](#_Toc97594660)

[**ГЛАВА 1. Проектирование структуры веб-приложения** 4](#_Toc97594661)

[1.1. Анализ предметной области 4](#_Toc97594662)

[1.2. Концептуальное проектирование 11](#_Toc97594663)

[1.3. Логическое проектирование 14](#_Toc97594664)

[1.4. Физическое проектирование 19](#_Toc97594665)

[**ГЛАВА 2. Описание функционирования веб-приложения** 25](#_Toc97594666)

[2.1. Назначение и перечень функций веб-приложения 25](#_Toc97594667)

[2.2. Программная реализация веб-приложения 26](#_Toc97594668)

[**Заключение** 52](#_Toc97594669)

[**Список используемых источников** 53](#_Toc97594670)

**Введение**

Объектом исследования курсового проекта является веб-приложение для автоматизации процессов продажи брендовой одежды через интернет-магазин.

Целью курсового проекта является разработка веб-приложения онлайн-магазина по продаже брендовой одежды на базе технологий ASP.NET Core Blazor Server.

Данная тема актуальна, так как в настоящее время интернет-торговля является одним из наиболее быстро развивающихся сегментов экономики. Рост популярности онлайн-покупок связан с удобством выбора товаров, возможностью сравнения цен, экономией времени и доступностью товаров независимо от географического расположения покупателя. Особенно это актуально для сферы продажи брендовой одежды, где покупателям важно иметь доступ к широкому ассортименту товаров от различных производителей, возможность фильтрации по брендам, категориям, размерам и ценам. Автоматизация процессов управления товарами, заказами и пользователями значительно повышает эффективность работы магазина и улучшает качество обслуживания клиентов [1].

Задачи курсового проекта:

1. Провести анализ предметной области, выполнить концептуальное, логическое и физическое проектирование веб-приложения онлайн-магазина по продаже брендовой одежды.

2. Разработать программу для автоматизации работы интернет-магазина с функциями каталога товаров, корзины покупок, системы заказов, авторизации пользователей и управления контентом.

# **ГЛАВА 1. Проектирование структуры веб-приложения**

## 1.1. Анализ предметной области

Поиск и изучение информации о потребностях заказчика, а также выяснение смысла данной информации для дальнейшей работы с проектом, называется анализом предметной области. Анализ предметной области – это первая фаза проектирования предметной области [2].

В качестве предметной области в данной курсовой работе рассматривается деятельность интернет-магазина по продаже брендовой одежды.

Выбор данной области был связан с потребностями рынка. С ростом объема информации обусловлено увеличение объема продаж, усложнение структуры ассортимента, необходимость управления большим количеством товарных позиций от различных брендов.

Каждому интернет-магазину нужен спрос на предоставляемые товары, поэтому в настоящий момент недостаточно просто продавать товар, чтобы удержаться на рынке. Сейчас важным фактором предприятия является современный веб-сайт с удобным интерфейсом для покупателей, предоставляющий возможность быстрого поиска нужных товаров, фильтрации по различным параметрам, оформления заказов и управления личным кабинетом. Также необходима административная панель для управления товарами, заказами и пользователями.

Реализация таких решений даёт возможность магазину увеличить покупательский спрос, прибыль и сэкономить время как клиентов, так и персонала магазина. Поэтому интернет-магазин брендовой одежды должен выполнять следующие основные функции:

* управление продажами (каталог товаров, корзина покупок, оформление заказов);
* управление товарным ассортиментом (добавление, редактирование, удаление товаров, брендов, категорий);
* управление заказами (просмотр, изменение статусов, обработка заказов);
* управление пользователями и кадровый учет (регистрация, авторизация, роли, личные кабинеты);
* информационная (обеспечение клиентов информацией о товарах, брендах, доступности).

Данные функции реализуются в большинстве современных интернет-магазинов.

В настоящее время существует множество готовых решений для создания интернет-магазинов. Рассмотрим несколько примеров успешных интернет-магазинов брендовой одежды.

Данные функции реализуются в большинстве разных производств.

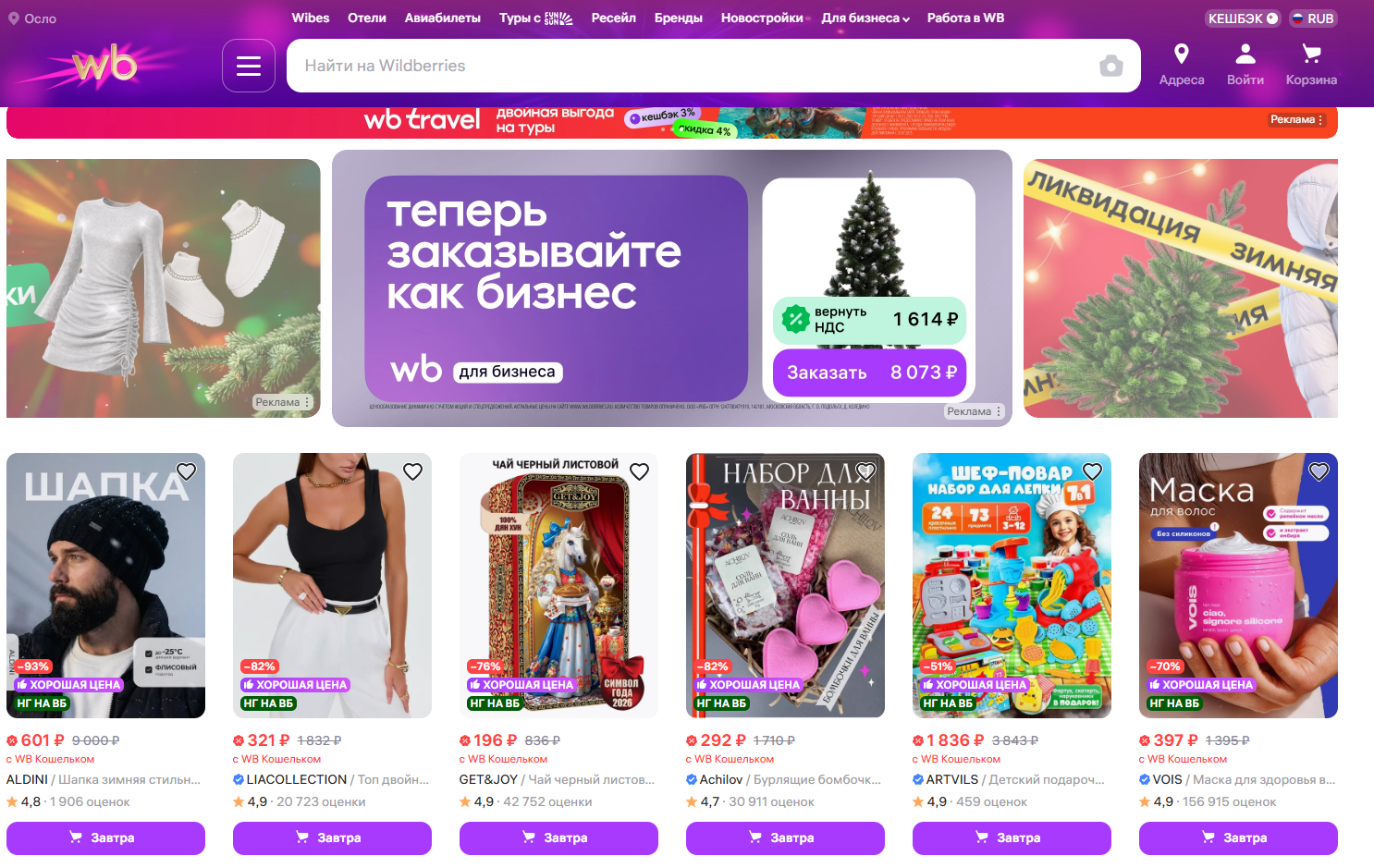
Одним из популярных примеров является сайт «Wildberries» (https://www.wildberries.ru/). Данный интернет-магазин предоставляет широкий ассортимент товаров различных категорий, включая брендовую одежду. На сайте реализованы функции каталога с фильтрацией по различным параметрам, корзины покупок, личного кабинета пользователя, отслеживания заказов. (рис. 1.1).

Рисунок 1.1 − Главная страница сайта «Wildberries»

Другим примером является сайт «Lamoda» (https://www.lamoda.ru/). Этот интернет-магазин специализируется на продаже модной одежды и обуви. На сайте реализованы функции каталога товаров с детальными описаниями, изображениями, размерами, фильтрации по брендам, категориям, размерам, цветам и ценам, корзины покупок, оформления заказов с выбором способа доставки, системы скидок и акций, личного кабинета с историей заказов. (рис. 1.2).

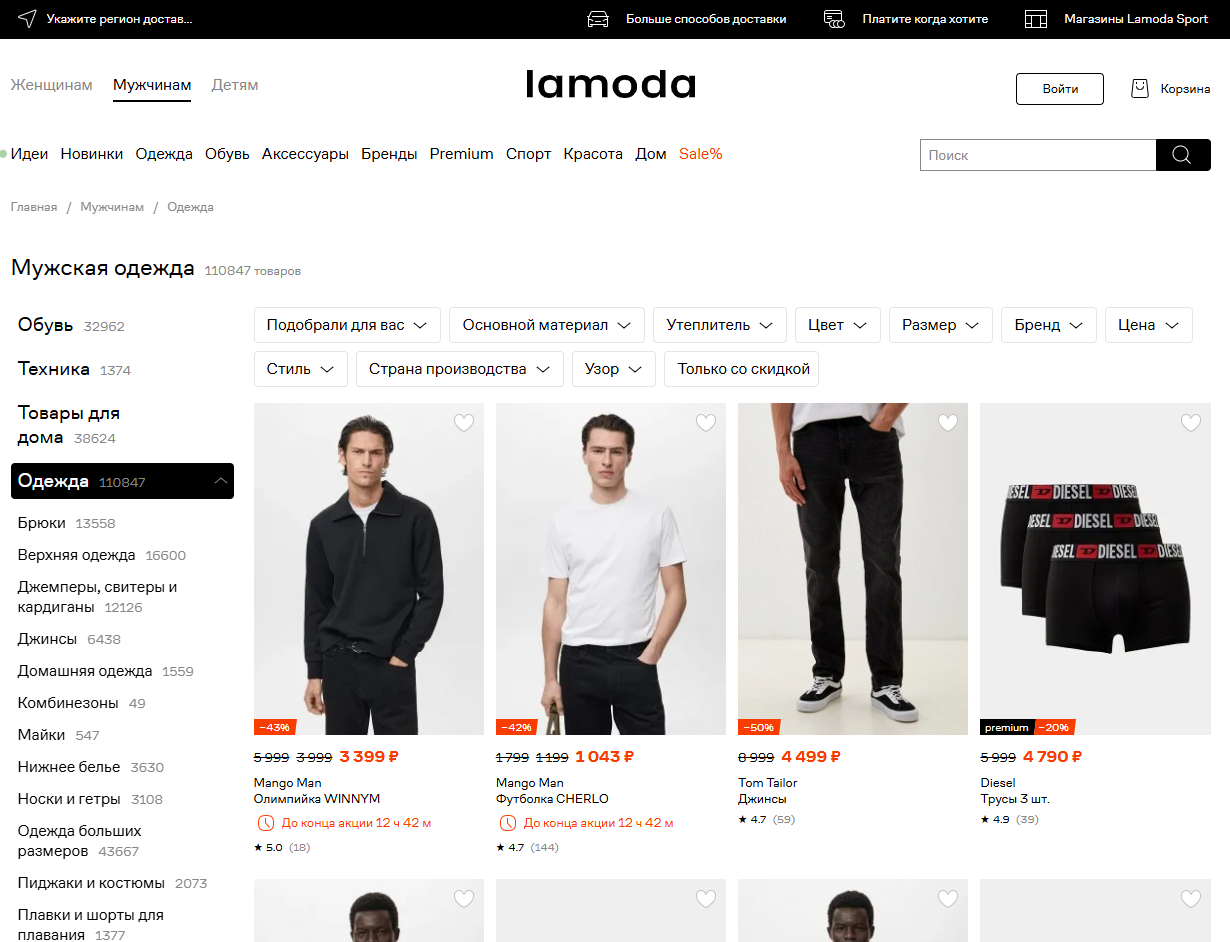


Рисунок 1.2 − Главная страница сайта «Lamoda»

Также можно рассмотреть интернет-магазин «Brandshop» (https://brandshop.ru/), который специализируется на продаже брендовой одежды и обуви. На сайте реализованы функции каталога с подробными карточками товаров, фильтрации по брендам и категориям, корзины покупок, системы авторизации и регистрации, управления заказами, программы лояльности. (рис. 1.3).

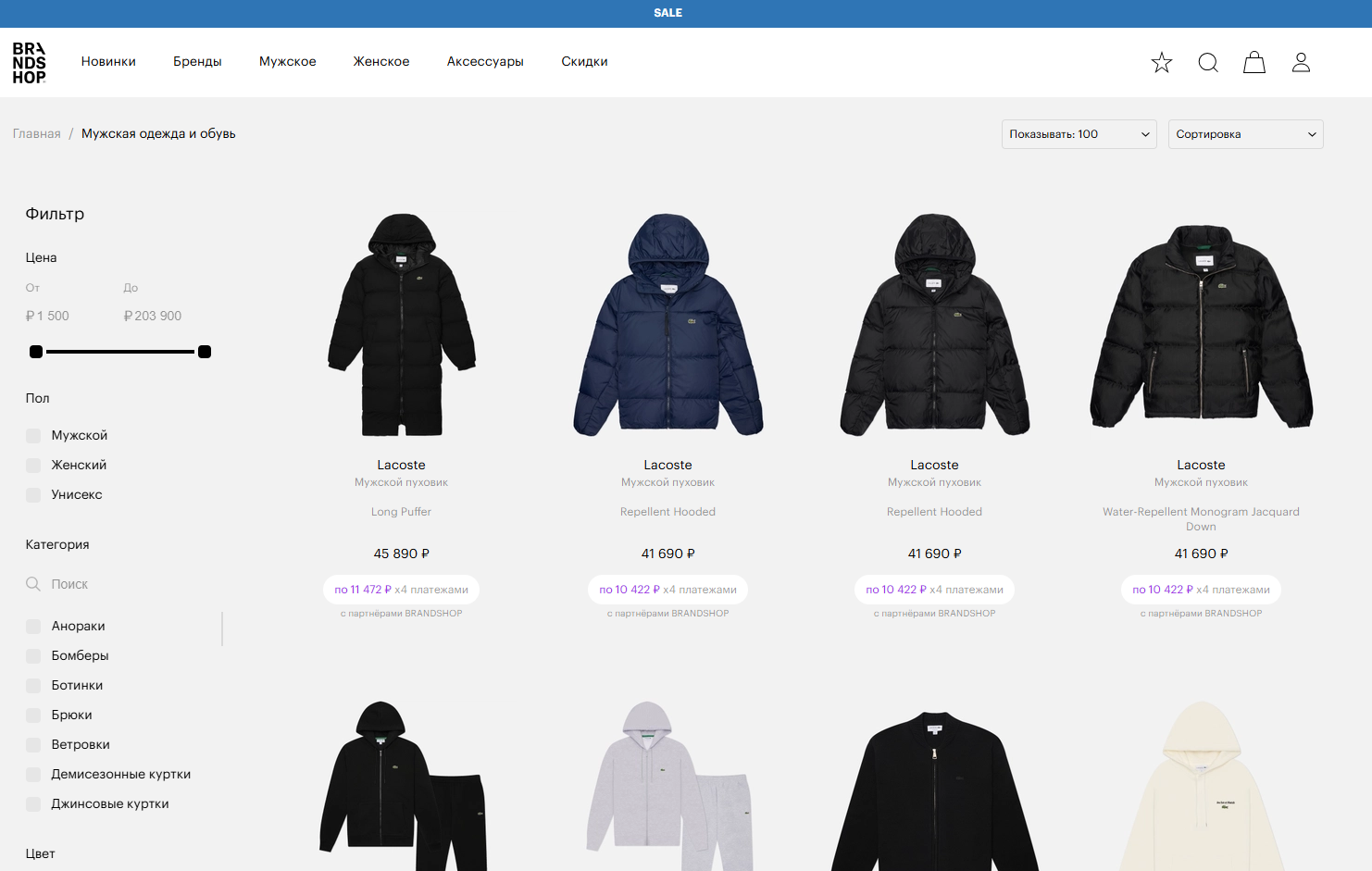


Рисунок 1.3 − Главная страница сайта «Brandshop»

При сравнении рассмотренных интернет-магазинов было выявлено, что все они имеют схожий функционал: каталог товаров с фильтрацией, корзину покупок, систему заказов, авторизацию пользователей. Основные отличия заключаются в дизайне интерфейса, дополнительных функциях (например, программа лояльности, системы рекомендаций) и масштабе ассортимента.

На основе проведённого анализа предметной области был выделен перечень функций, которые будут реализованы в курсовом проекте:

1. Каталог товаров с фильтрацией по брендам, категориям и поиском по названию.
2. Детальный просмотр информации о товаре.
3. Корзина покупок с возможностью изменения количества товаров.
4. Оформление заказов с указанием адреса доставки.
5. Система авторизации и регистрации пользователей с дополнительными полями (пол, дата рождения, предпочитаемый размер, город).
6. Система ролей и разграничение прав доступа (Администратор, Менеджер, Покупатель).
7. Административная панель для управления товарами (CRUD-операции).
8. Административная панель для управления брендами и категориями (CRUD-операции).
9. Административная панель для управления заказами.
10. Экспорт и импорт данных в формате Excel.
11. Личный кабинет пользователя с историей заказов.

## 1.2. Концептуальное проектирование

Концептуальное проектирование заключается в формализованном описании предметной области, которое должно быть таким, чтобы с одной стороны, можно было проанализировать корректность схемы БД, а с другой стороны не должно быть привязано к конкретной СУБД.

Любая БД создаётся для решения определенных прикладных задач: в простейшем случае – накопление и выдача данных, в более сложном – решение задач, использующих данные из БД в качестве исходных данных.

Для концептуального проектирования используется диаграмма вариантов использования.

Диаграмма вариантов использования (Use Case Diagram) состоит из графической диаграммы, описывающей действующие лица и прецеденты, а также спецификации, представляющей собой текстовое описание конкретных последовательностей действий (потока событий), которые выполняет пользователь при работе с системой.

Главное назначение диаграммы вариантов использования – формализация функциональных требований к системе с помощью действующих лиц (потенциальных пользователей системы) и вариантов использования (задачи, с которыми действующее лицо обращается к разрабатываемому продукту).

На диаграмме использования изображаются:

* Действующие лица – группы лиц или систем, взаимодействующих с нашей системой;
* Варианты использования (прецеденты) – сервисы, которые наша система предоставляет;
* Отношения между элементами диаграммы.

Выделим свойства и структуру базы данных для автоматизации интернет-магазина брендовой одежды.

Действующие лица:

* Администратор – имеет полный доступ ко всем функциям системы, управляет товарами, брендами, категориями, заказами и пользователями;
* Менеджер – управляет заказами, просматривает статистику, имеет доступ к управлению товарами;
* Покупатель (Customer) – просматривает каталог, добавляет товары в корзину, оформляет заказы, управляет личным кабинетом.

Рассмотрим варианты использования для каждого действующего лица.

**Администратор:**

1. Управление товарами (добавление, редактирование, удаление товаров).
2. Управление брендами (добавление, редактирование, удаление брендов).
3. Управление категориями (добавление, редактирование, удаление категорий).
4. Управление заказами (просмотр всех заказов, изменение статусов заказов).
5. Экспорт и импорт данных в Excel.
6. Просмотр информации о пользователях.
7. Управление ролями пользователей.

**Менеджер:**

1. Просмотр и управление заказами (изменение статусов заказов).
2. Просмотр каталога товаров.
3. Просмотр статистики по заказам.

**Покупатель:**

1. Просмотр каталога товаров.
2. Поиск товаров по названию.
3. Фильтрация товаров по брендам и категориям.
4. Просмотр детальной информации о товаре.
5. Добавление товаров в корзину.
6. Управление корзиной (изменение количества, удаление товаров).
7. Оформление заказа.
8. Просмотр истории своих заказов.
9. Регистрация и авторизация в системе.
10. Управление профилем пользователя.

Результатом этого этапа проектирования является построение первичной информационной структуры базы данных, которая называется концептуальной схемой базы данных или инфологической моделью.

Концептуальная схема базы данных содержит сгруппированные атрибуты предметной области по признакам функциональной зависимости.

В первом приближении для решения выделенных задач необходимо хранение данных о следующих объектах:

1. Товар (Product) – информация о товарах магазина.
2. Бренд (Brand) – информация о брендах одежды.
3. Категория (Category) – информация о категориях товаров.
4. Заказ (Order) – информация о заказах покупателей.
5. Элемент заказа (OrderItem) – информация о товарах, входящих в заказ.
6. Пользователь (ApplicationUser) – информация о пользователях системы (расширение стандартной модели Identity).

## 1.3. Логическое проектирование

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных логическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

При создании моделей данных используется метод семантического моделирования. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты диаграмм “сущность-связь” [7].

Существуют различные варианты отображения ER-диаграммы, но все варианты диаграмм “сущность-связь” исходят из одной идеи – рисунок всегда нагляднее текстового описания. ER-диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями [6].

Основные преимущества ER-моделей:

* наглядность;
* модели позволяют проектировать базы данных с большим количеством объектов и атрибутов;
* ER-модели реализованы во многих системах автоматизированного проектирования баз данных.

Основные элементы ER-моделей:

* объекты (сущности);
* атрибуты объектов;
* связи между объектами.

Сущность – объект предметной области, имеющий атрибуты.

Связь между сущностями характеризуется:

* типом связи (1:1, 1:N, N:М);
* классом принадлежности.

Класс может быть обязательным и необязательным. Если каждый экземпляр сущности участвует в связи, то класс принадлежности – обязательный, иначе –необязательный.

Результатом логического проектирования является логическая модель данных, состоящая из ER-диаграммы или диаграммы отношений, а также из реляционной схемы.

Рассмотрим ER-диаграмму “Клиент оставляет заказ” (рис. 1.12).

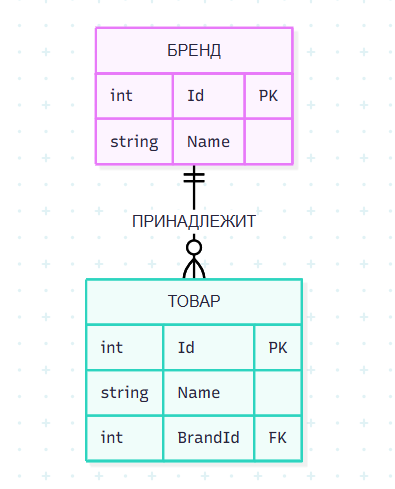


Рисунок 1.12 – ER-диаграмма “Клиент оставляет заказ”

Связь ОСТАВЛЯЕТ имеет тип 1:М, так как клиент может оставить несколько заказов. Сущность КЛИЕНТ имеет обязательный класс принадлежности, так как полагаем, что каждый клиент обязательно оставит заказ. Сущность ЗАКАЗ имеет обязательный класс принадлежности, так как каждый заказ имеет своего клиента.

Рассмотрим ER-диаграмму “Сотрудник формирует заказ” (рис. 1.13).

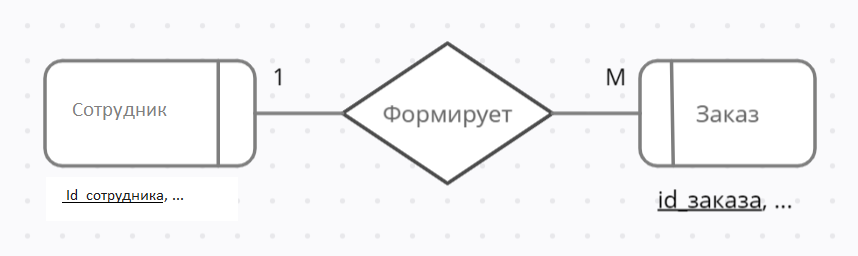


Рисунок 1.13 – ER-диаграмма “Продавец формирует заказ”

Связь ФОРМИРУЕТ имеет тип 1:М, так как продавец может формировать несколько заказов. Сущность СОТРУДНИК имеет обязательный класс принадлежности, так как полагаем, что каждый сотрудник обязательно оформляет заказ. Сущность ЗАКАЗ имеет обязательный класс принадлежности, так как каждый заказ имеет своего продавца.

Рассмотрим ER-диаграмму “Заказ состоит из изделий” (рис. 1.14)



Рисунок 1.14 – ER-диаграмма “Заказ состоит из изделий”

Связь СОСТОИТ ИЗ имеет тип М:М, так как во многие заказы могут входить одинаковые по категории изделия. Сущность ЗАКАЗ имеет обязательный класс принадлежности, так как в каждом заказе есть хотя бы одно изделие. Сущность ИЗДЕЛИЕ имеет необязательный класс принадлежности, так как полагаем, что не каждое изделие может входить в заказ.

Рассмотрим ER-диаграмму “Изделие шьётся из ткани” (рис. 1.15)



Рисунок 1.15 – ER-диаграмма “Изделие шьётся из ткани”

Связь ШЬЁТСЯ ИЗ имеет тип М:1, так как считаем, что несколько изделий могут шиться из одной ткани. Сущность ИЗДЕЛИЕ имеет обязательный класс принадлежности, так как полагаем, что каждое ИЗДЕЛИЕ шьется из ткани. Сущность ТКАНЬ имеет необязательный класс принадлежности, так как считаем, что не каждая ТКАНЬ может использоваться в изделиях.

Рассмотрим ER-диаграмму “Поставщик осуществляет поставку” (рис. 1.16).

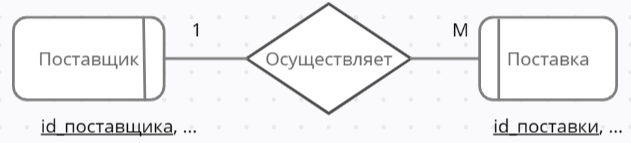


Рисунок 1.16 – ER-диаграмма “Поставщик осуществляет поставку”

Связь ОСУЩЕСТВЛЯЕТ имеет тип 1:М, так как один поставщик может осуществлять несколько поставок. Сущность ПОСТАВКА имеет обязательный класс принадлежности, так как полагаем, что каждая поставка осуществляется поставщиком. Сущность ПОСТАВЩИК имеет обязательный класс принадлежности, так как каждый поставщик осуществляет хотя бы одну поставку.

Рассмотрим ER-диаграмму “Поставка состоит из тканей” (рис. 1.17)



Рисунок 1.17 – ER-диаграмма “Поставка состоит из тканей”

Связь СОСТОИТ ИЗ имеет тип 1:М, так как в одной поставке может находиться несколько тканей. Сущность ПОСТАВКА имеет обязательный класс принадлежности, так как в одной поставке есть хотя бы одна ткань. Сущность ТКАНЬ имеет необязательный класс принадлежности, так как полагаем, что не каждая ТКАНЬ входит в поставку.

Формирование отношений на основе диаграмм ER-типа

Правило 1. Если степень связи 1:1 и класс принадлежности обеих сущностей обязательный, то требуется всего одно отношение, объединяющее эти сущности. При этом ключом этого отношения может быть ключ любой сущности.

Правило 4. Если степень связи 1:M (M:1) и класс принадлежности М-связной сущности − обязательный, то, независимо от КП другой сущности, достаточно сформировать два отношения: по одному для каждой сущности с соответствующими ключами. При этом ключ односвязной сущности добавляется в качестве атрибута в отношение для многосвязной сущности.

Правило 5. Если степень связи 1:M (M:1) и класс принадлежности М-связной сущности – необязательный, то необходимо формирование трех отношений: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.

Правило 6. Если степень связи M:M, то необходимо формирование трех отношений: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.

Сформируем набор предварительных отношений на основе ER-диаграмм с указанием первичного ключа для каждого отношения:

1. Связь “Клиент оставляет заказ” (рис. 1.12), по правилу №4 формируем два отношения:
   * Клиент (id\_клиента, …);
   * Заказ (id\_заказа, id\_клиента, …).
2. Связь “Сотрудник формирует заказ” (рис. 1.13), по правилу №4 формируем два отношения:
   * Сотрудник (id\_сотрудника, …);
   * Заказ (id\_заказа, id\_сотрудника, …).
3. Связь “Заказ состоит из изделий” (рис. 1.14), по правилу №6 формируем два отношения:

* Заказ (id\_заказа, …);
* Изделие (id\_изделия, …);
* СоставЗаказа (id\_заказа, id\_изделия).

1. Связь “Изделие шьётся из ткани” (рис. 1.15), по правилу №4 формируем два отношения:
   * Изделие (id\_изделия, id\_ткани …);
   * Ткань (id\_ткани, …);
2. Связь “Поставщик осуществляет поставку” (рис. 1.16), по правилу №4 формируем два отношения:
   * Поставщик (id\_поставщика, …);
   * Поставка (id\_поставки, id\_поставщика, …).
3. Связь “Поставка состоит из тканей” (рис. 1.17), по правилу №5 формируем два отношения:
   * Поставка (id\_поставки, …);
   * Ткань (id\_ткани, …);
   * СоставПоставки (id\_поставки, id\_ткани, …).

Набор предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения:

1. Сотрудник (id\_сотрудника, имя, фамилия, дата рождения, зарплата, стаж).
2. Заказ (id\_заказа, id\_клиента, id\_сотрудника, количество, цена, дата оформления, дата выдачи).
3. Изделие (id\_изделия, id\_ткани, название, требования по уходу, цвет, цена).
4. Ткань (id\_ткани, название, состав, цвет, длина, ширина, тип).
5. Клиент (id\_клиента, имя, фамилия, номер телефона).
6. Поставщик (id\_поставщика, имя, фамилия, номер телефона, адрес).
7. Поставка (id\_поставки, id\_поставщика, дата, время).
8. СоставПоставки (id\_поставки, id\_ткани, количество, вес).

## 1.4. Физическое проектирование

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

На этапах концептуального и логического проектирования не принимаются во внимание конкретные возможности целевой СУБД.

В качестве СУБД для данной самостоятельной работы был выбран Microsoft SQL Server 2018. Среда SQL Server Management Studio (SSMS) – это графический набор средств для разработки сценариев на T-SQL и управления всеми компонентами SQL Server. Management Studio является основным инструментом любого разработчика или администратора MS SQL сервера [4]. Перечислим основные возможности среды SQL Server Management Studio:

* Подключение к любому компоненту SQL Server.
* Обозреватель объектов.
* Создание и редактирование сценариев.
* Настройка свойств сервера, баз данных и других объектов.

На этапе логического проектирования были разработаны ER-диаграммы. Далее нужно на их основе определить требования к структуре таблиц и физически реализовать таблицы в СУБД (см. табл. 1.1 – 1.11).

Таблица 1.1 – Структура таблицы «Сотрудник»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_сотрудника | Целое | NOT NULL | Первичный | — | — | — |
| Имя | Строка | NOT NULL | — | — | — | — |
| Фамилия | Строка | NOT NULL | — | — | — | — |
| Дата рождения | Дата | NOT NULL | — | — | — | — |
| Зарплата | Целое | NOT NULL | — | — | >20000 | — |
| Стаж | Строка | NOT NULL | — | — | — | — |

Таблица 1.2 – Структура таблицы «Заказ»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_заказа | Целое | NOT NULL | Первичный | — | — | — |
| id\_клиента | Целое | NOT NULL | Внешний | — | — | id\_клиента в таблице «Клиент» |
| Количество | Целое | NOT NULL | — | — | >0 | — |
| Цена | Целое | NOT NULL | — | — | — | — |
| Дата оформления | Дата | NOT NULL | — | Текущая | — | — |
| Дата выдачи | Дата | NOT NULL | — | — | — | — |

Таблица 1.3 – Структура таблицы «Изделие»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_изделия | Целое | NOT NULL | Первичный | — | — |  |
| id\_ткани | Целое | NOT NULL | Внешний | — | — | id\_ткани в таблице «Ткань» |
| Название | Строка | NOT NULL | — | — | — | — |
| Требования по уходу | Строка | NOT NULL | — | — | — | — |
| Цвет | Строка | NOT NULL | — | — | — | — |
| Цена | Целое | NOT NULL | — | — | >0 | — |

Таблица 1.4 ­– Структура таблицы «Ткань»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_ткани | Целое | NOT NULL | Первичный | — | — | — |
| Название | Строка | NOT NULL | — | — | — | — |
| Цвет | Строка | NOT NULL | — | — | — | — |
| Длина | Целое | NOT NULL | — | — | >0 | — |
| Ширина | Целое | NOT NULL | — | — | >0 | — |
| Тип | Строка | NOT NULL | — | — | — | — |

Таблица 1.5 – Структура таблицы «Клиент»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_клиента | Целое | NOT NULL | Первичный | — | — | — |
| Имя | Строка | NOT NULL | — | — | — | — |
| Фамилия | Строка | NOT NULL | — | — | — | — |
| Номер телефона | Строка | NOT NULL | — | — | — | — |

Таблица 1.6 – Структура таблицы «Поставщик»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_поставщика | Целое | NOT NULL | Первичный | — | — | — |
| Имя | Строка | NOT NULL | — | — | — | — |
| Фамилия | Строка | NOT NULL | — | — | — | — |
| Номер телефона | Строка | NOT NULL | — | — | — | — |
| Адрес | Строка | NOT NULL | — | — | — | — |

Таблица 1.7 – Структура таблицы «Поставка»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_поставки | Целое | NOT NULL | Первичный | — | — | — |
| Id\_поставщика | Целое | NOT NULL | Внешний | — | — | Id\_поставщика в таблице «Поставщик» |
| Дата | Дата | NOT NULL | — | — | — | — |
| Время | Время | NOT NULL | — | — | — | — |

Таблица 1.8 – Структура таблицы «СоставПоставки»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_поставки | Целое | NOT NULL | Первичный/Внешний | — | — | id\_поставки в таблице «Поставка» |
| id\_ткани | Целое | NOT NULL | — | — | id\_ткани в таблице «Ткань» |
| Количество | Целое | NOT NULL | — | — | >20 | — |
| Вес | Целое | NOT NULL | — | — | — | — |

Свяжем таблицы и создадим схему баз данных, содержащую эти таблицы (рис. 1.22)

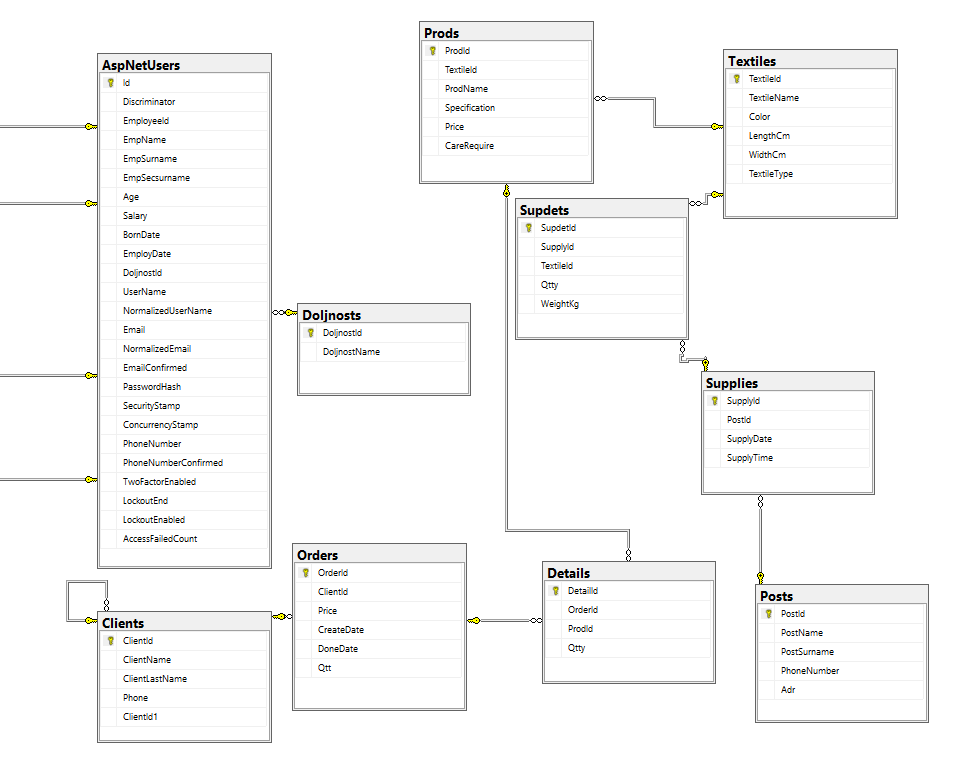


Рисунок 1.22 – Схема данных

Диаграмма IDEF0 (Integration Definition for Function Modeling) является инструментом визуализации, использующимся для моделирования функциональных процессов в системе или организации. Она позволяет графически представлять различные виды бизнес-процессов, производственных операций и других видов предприятий на любом уровне детализации.

Диаграмма IDEF0 состоит из двух основных элементов: функций (или процессов) и концепций. Функции представляются в виде прямоугольников (или "боксов"), а концепции представляются в виде стрелок, входящих в прямоугольник (входы и контроли), или выходящих из него (выходы и механизмы).

 Составим диаграмму IDEF0 для представления и анализа функций, выполняемых веб-приложением, и для записи механизмов, с помощью которых эти функции выполняются (рис. 1.23-1.24)

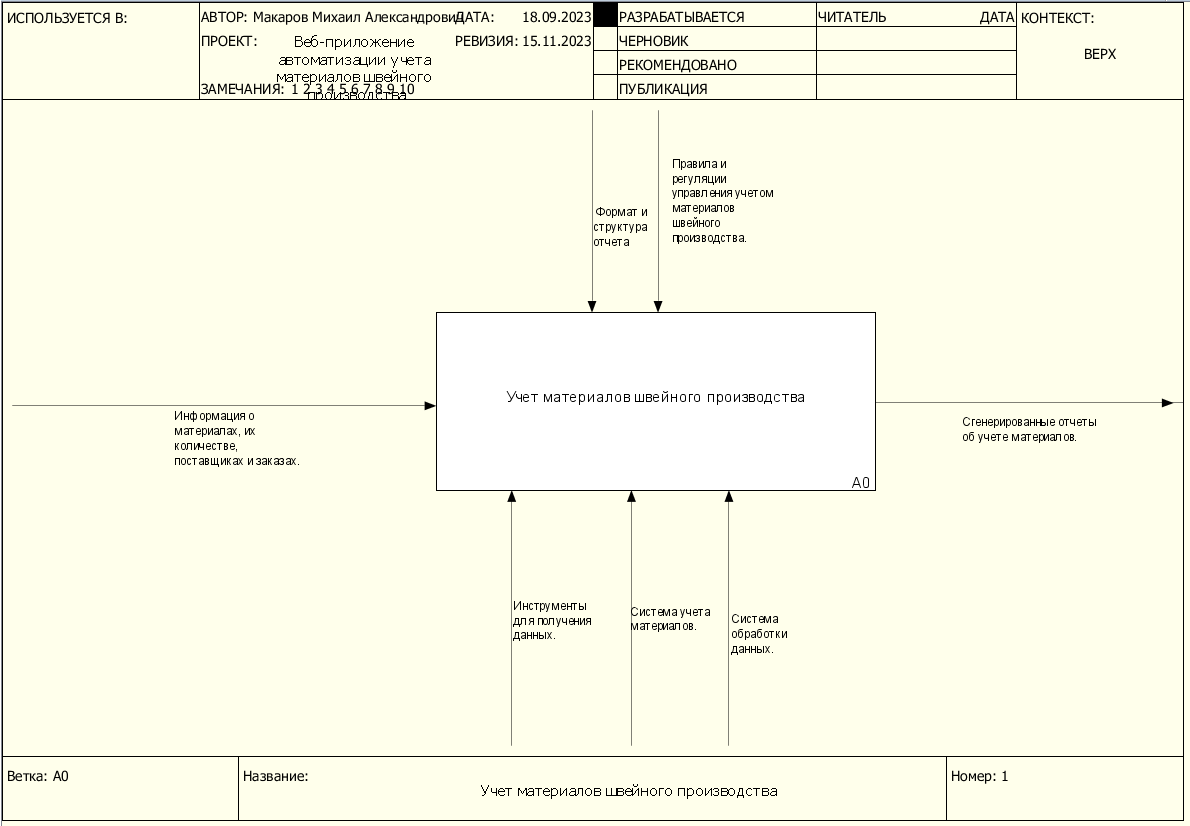


Рисунок 1.23 – Диаграмма верхнего уровня A0

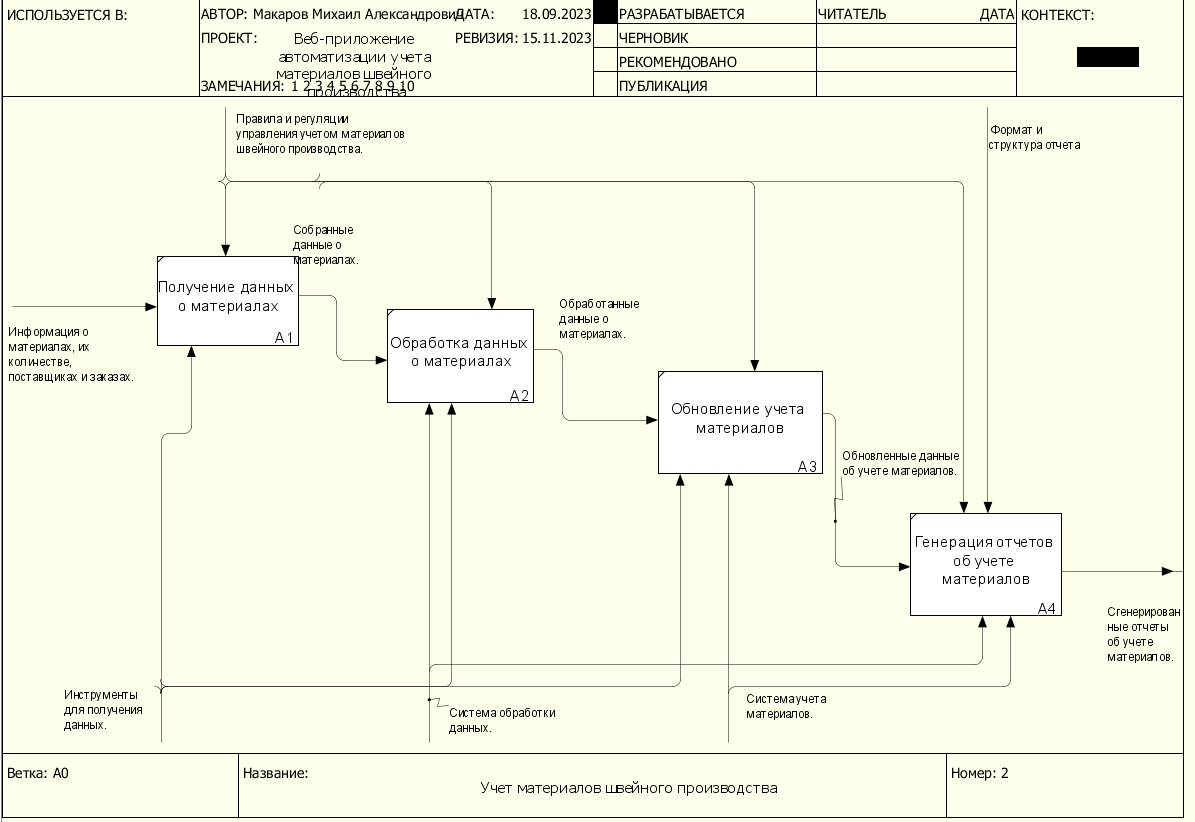


Рисунок 1.24 – Декомпозиция диаграммы верхнего уровня

Выводы

Рассмотрены концепции БД. Сформулированы требования к информации в БД «Швейное производство».

# **ГЛАВА 2. Описание функционирования** **веб-приложения**

## 2.1. Назначение и перечень функций веб-приложения

В процессе выполнения курсового проекта было создано веб-приложение автоматизации учета материалов и разработан её интерфейс для удобства в работе пользователей с данными о заказах, клиентах, продукции, поставках, деталях поставок и сотрудниках.

При реализации функционала данной базы данных использованы СУБД MS SQL Server для создания базы данных и для создания веб приложения я используется ASP.NET Core. ASP.NET Core — свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным кодом. Функции веб приложения:

* добавление, редактирование и хранение информации о клиентах;
* добавление, редактирование, удаление и хранение информации о заказах клиентов;
* экспорт и импорт данных, добавление, редактирование, удаление и хранение информации о составе заказов;
* экспорт и импорт данных, добавление, редактирование, удаление и хранение информации о продукции;
* добавление, редактирование и хранение информации о поставках;
* добавление, редактирование, удаление и хранение информации о деталях поставки;
* внесение информации о сотрудниках;
* экспорт и импорт данных, отчёт о имеющейся продукции с информацией о использующихся тканях в них.

## 2.2. Программная реализация веб-приложения

Работа над программой начинается с установки необходимых пакетов поставщика. (рис. 2.1).

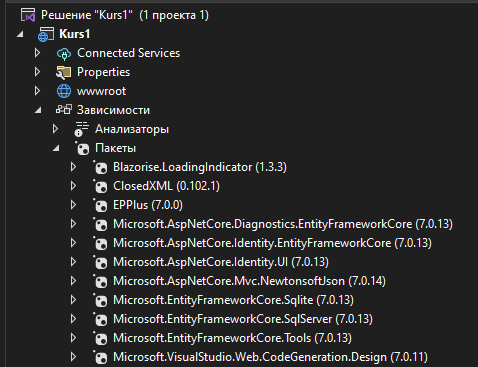


Рисунок 2.1 − Пакеты поставщика

Создаём модели данных. В папке Models создаём необходимый класс, содержащий следующий код: (рис. 2.2).

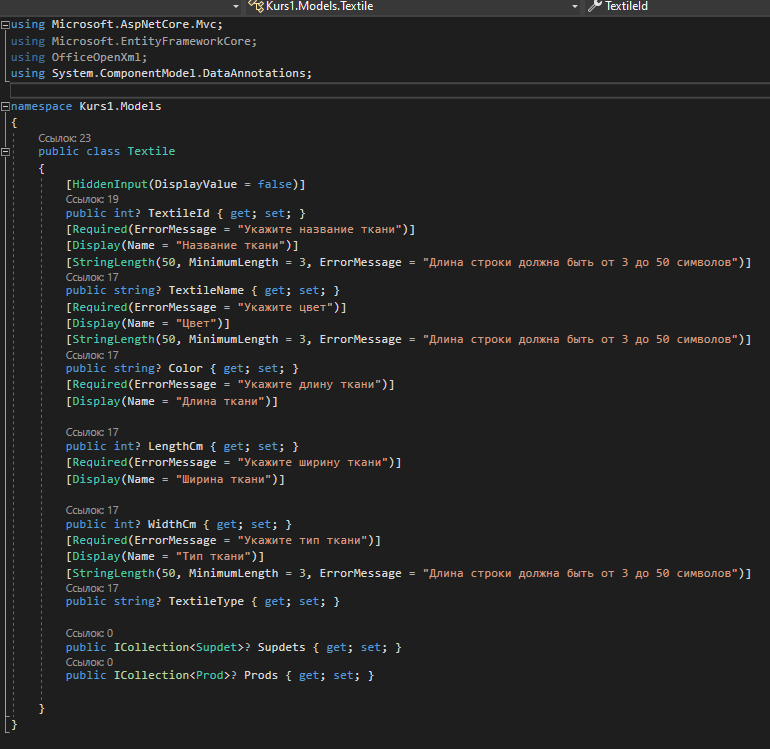


Рисунок 2.2 − Фрагмент экранной копии, показывающий модель Textile

Аналогичным образом создаём оставшиеся модели данных (рис. 2.3 ).

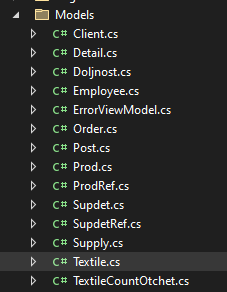


Рисунок 2.3 − Фрагмент экранной копии, показывающий модели данных

После создания моделей нужно обратиться к контексту базы данных. Контекст базы данных DbContext — это основной класс, который координирует функциональные возможности EF для определённой моделиданных. Этот класс является производным от класса Microsoft.EntityFrameworkCore.DbContext. Производный класс DbContext указывает сущности, которые включаются в модель данных. Некоторые расширения функциональности EF можно настроить. В проектес оответствующий класс называется ApplicationDbContext. Данный класс автоматический создаётся в папке Data, который необходимо заполнить следующим кодом (рис. 2.4).

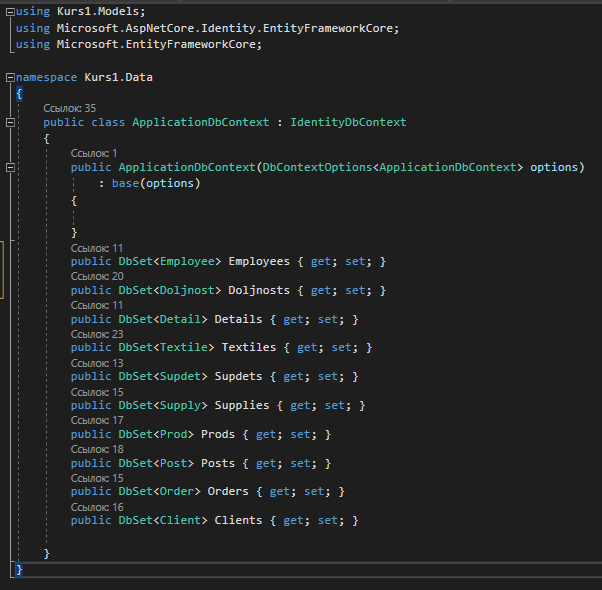


Рисунок 2.4 – Фрагмент экранной копии, класс ApplicationDbContext

Создаём контроллеры и представления. Используем подсистему формирования шаблонов Visual Studio для добавления контроллера и представлений MVC, которые будут использовать платформу EF для запроса данных и их сохранения. (рис. 2.5-2.7).

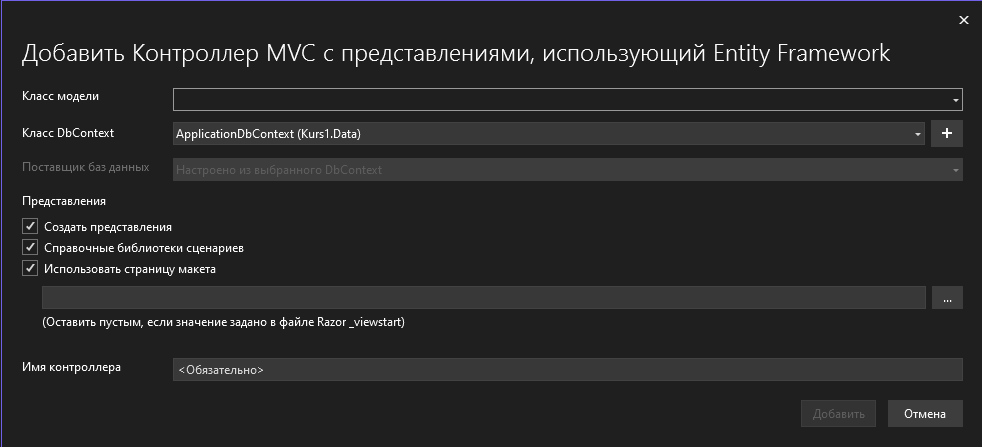


Рисунок 2.5 – Фрагмент экранной копии, форма создания контроллера.

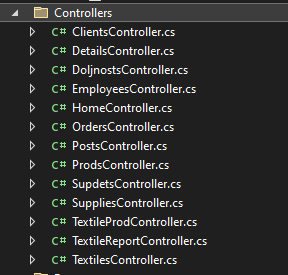


Рисунок 2.6 – Фрагмент экранной копии, отображающий список созданных контроллеров

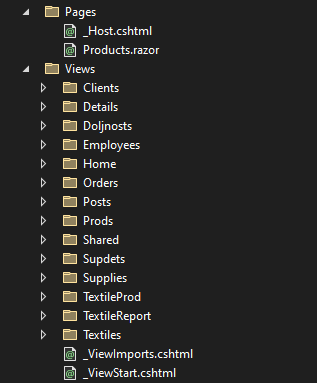


Рисунок 2.7 – Фрагмент экранной копии, отображающий список созданных представлений

Далее необходимо выполнить миграцию и обновить базу данных. Для этого поблючаем базу данных. Переходим в во вкладку appsettings.json и прописываем путь базы данных. (рис. 2.8).

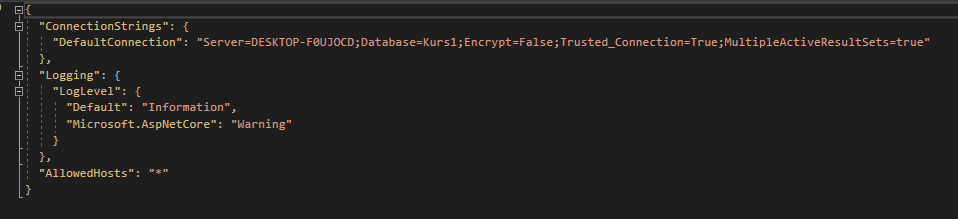


Рисунок 2.8 – Фрагмент экранной копии, показывающий строку подключения базы данных

Выполняем миграцию. Переходим в консоль диспетчера пакетов и вводим команду Add-Migration migr1, затем вводим команду Update-Database для обновления данных. (рис. 2.9 – 2.10).

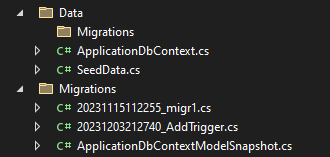


Рисунок 2.9 – Фрагмент экранной копии, показывающий созданные миграции



Рисунок 2.10 – Фрагмент экранной копии, показывающий созданную базу данных и её структуру

Проверяем правильность заполнение моделей с помощью диаграммы классов. Для этого переходим в раздел “Диаграммы баз данных” и создаём диаграмму (рис. 2.11).

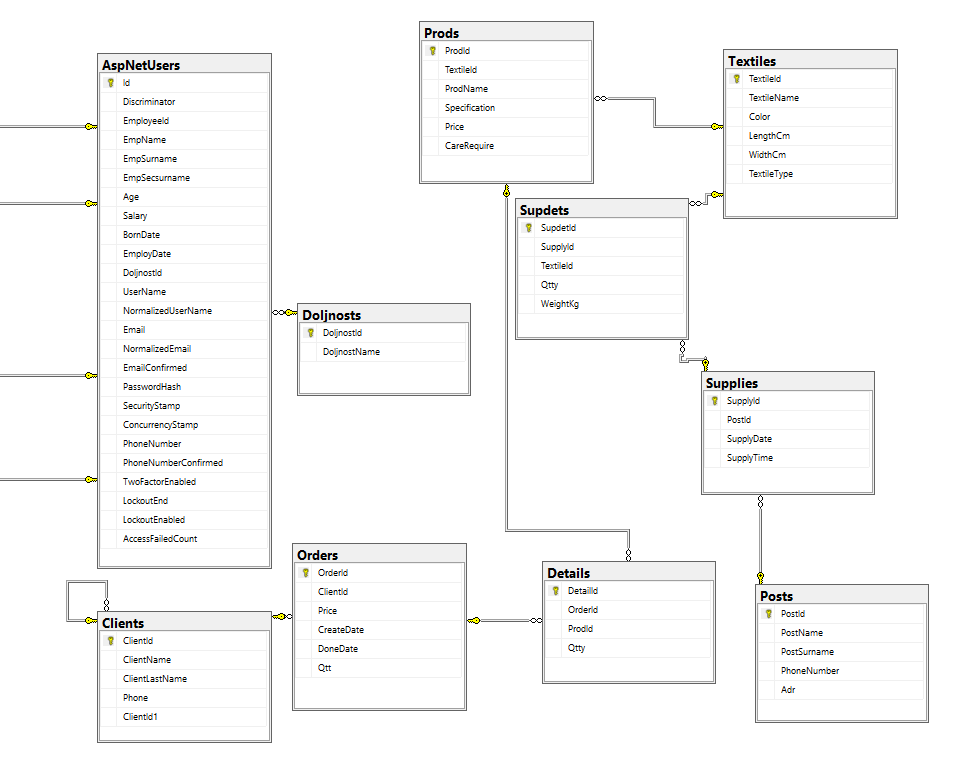


Рисунок 2.11 – Фрагмент экранной копии, диаграмма базы данных

Чтобы созданные элемменты были видны нужно организовать переходы между страницами. В папке с представлениями изменяем код \_Layout.cshtml (рис. 2.12-2.13).



Рисунок 2.12 – Фрагмент экранной копии, демонстрирующий код \_Layout.cshtml



Рисунок 2.13 – Фрагмент экранной копии, демонстрирующий результат отображения

Добавление триггера в базу данных с помощью миграции. Триггер реализует проверку вводимых данных. Сначала необходимо создаеть новую миграцию с именем AddTrigger в пространстве имен Kurs1.Migrations. Внутри этого класса, переопределяем два метода - Up и Down.

Метод Up используется для определения логики, которая будет применена при применении миграции. В данном случае используем метод migrationBuilder.Sql для выполнения SQL-скрипта, который создает новый триггер в базе данных.

Прописываем триггер с названием Check\_Textile, он создается для таблицы Textiles и активируется вместо операции INSERT. Он проверяет, что значения LengthCm и WidthCm вставляемой записи больше 0. Если это условие выполняется, то запись добавляется в таблицу. В противном случае транзакция откатывается и генерируется ошибка.

Метод Down используется для определения логики, которая будет применена при откате миграции. В данном случае удаляет созданный ранее триггер из базы данных.

Таким образом, добавление этого триггера позволяет обеспечить дополнительную проверку данных на уровне базы данных, что улучшает надежность и безопасность приложения(рис. 2.14-2.15).



Рисунок 2.14 – Фрагмент экранной копии, показывающий созданую миграцию AddTrigger и код триггера Check\_Textile

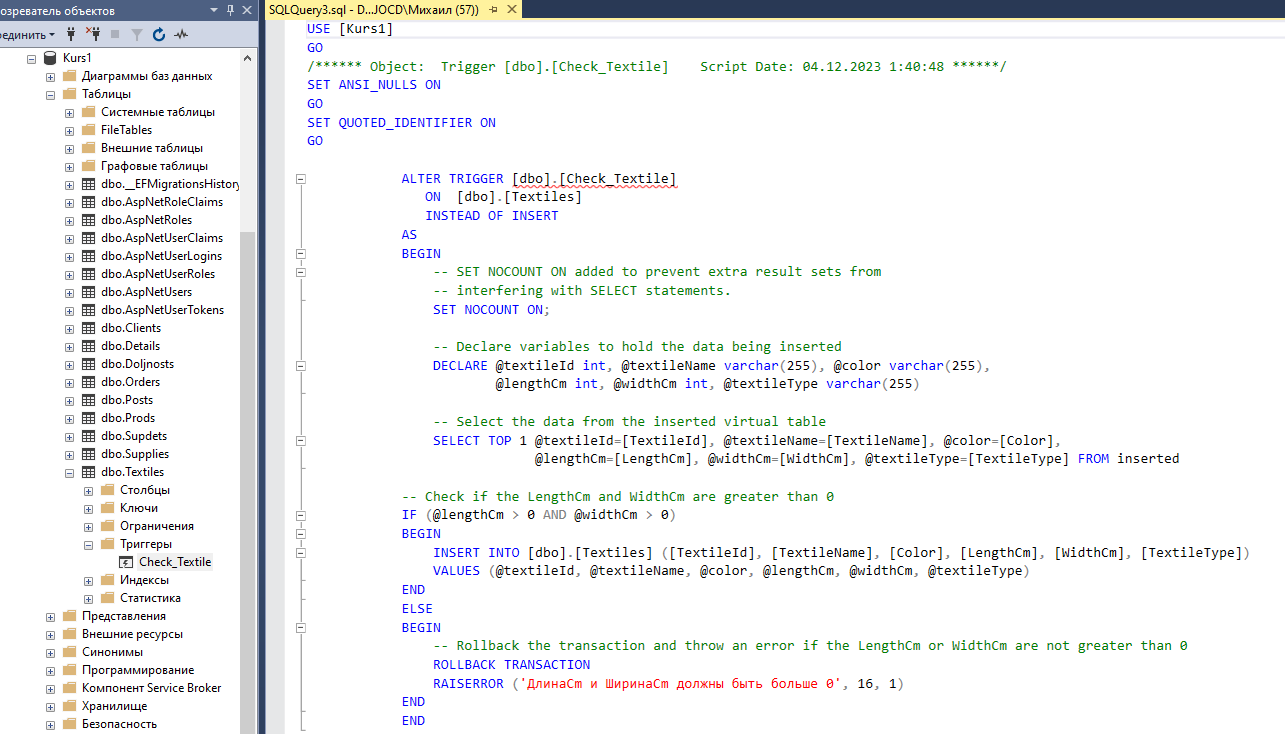


Рисунок 2.15 – Фрагмент экранной копии, результат миграциии

Удостоверение и вход в систему (Authentication): Добавление функциональности для входа в систему, используя стандартные контроллеры и представления, предоставляемые ASP.NET Core Identity. Это включает в себя страницы входа и регистрации, а также обработку идентификации пользователей (рис. 2.16-2.17).

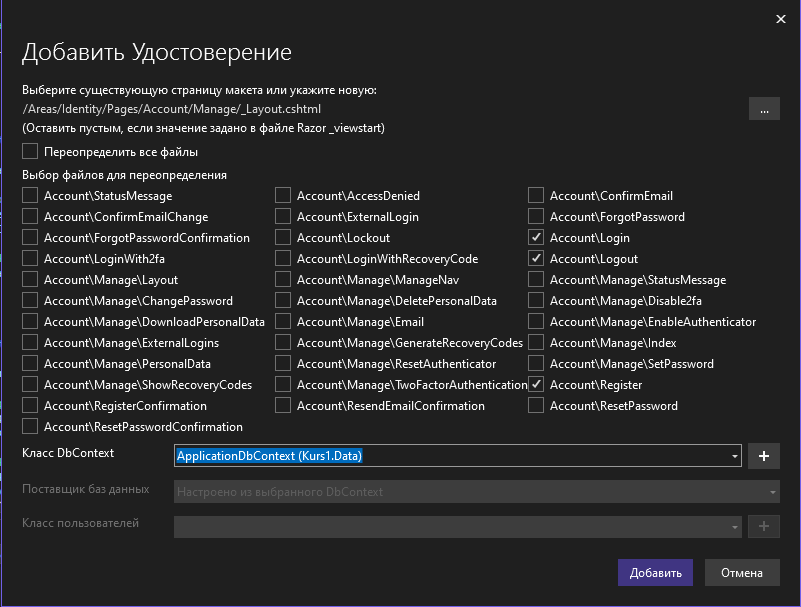


Рисунок 2.16 – Фрагмент экранной копии, форма добавления удостоверения

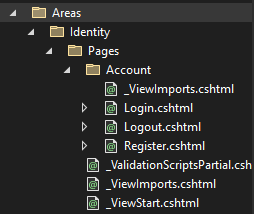


Рисунок 2.17 – Фрагмент экранной копии, результат добавления страниц авторизации

Дополнительные поля при регистрации: Добавление дополнительных полей при регистрации пользователя, такие как ФИО, возраст, должность. Это достигается путем обновления представлений регистрации и управления учетной записью (рис. 2.18).

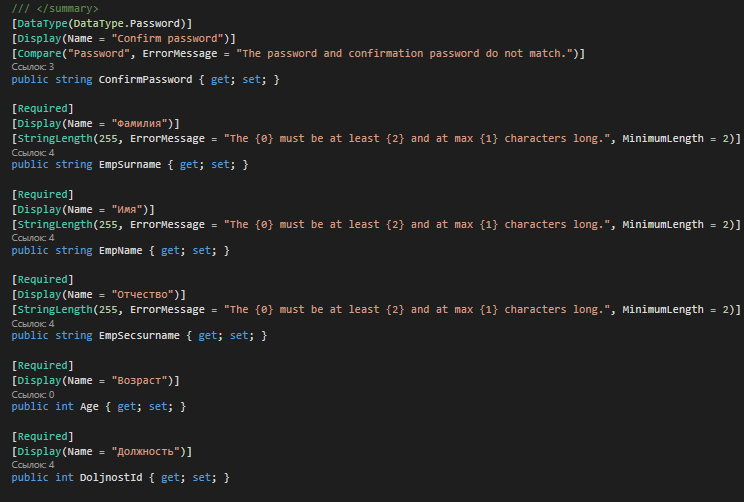


Рисунок 2.18 – Фрагмент экранной копии, добавление новых полей регистрации

Роли и политики (Authorization): Система ролей и политик, которая позволяет управлять доступом пользователей к различным частям вашего приложения. Это достигается путем использования атрибута [Authorize] и его вариаций для ограничения доступа к определенным контроллерам и действиям (рис. 2.19).

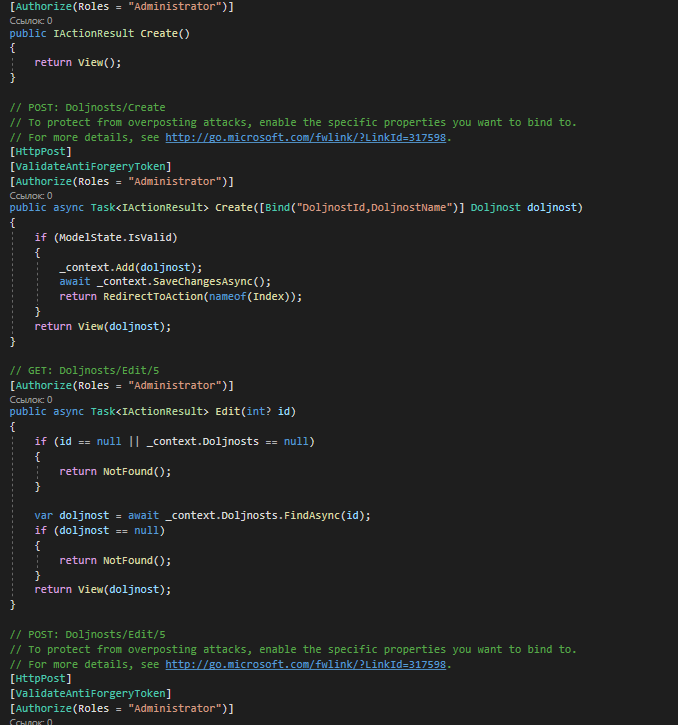


Рисунок 2.19 – Фрагмент экранной копии, использование атрибута “[Authorize]”

Разграничение прав доступа: система разграничения прав доступа, которая позволяет определить, какие действия может выполнять каждый пользователь в зависимости от его роли. Это достигается путем создания пользовательских атрибутов авторизации, которые проверяют роль пользователя перед выполнением действия (рис. 2.20).

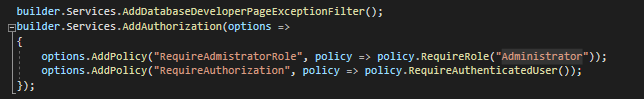


Рисунок 2.20 – Фрагмент экранной копии, добавление политики

Реализация инициализация данных при первом запуске приложения. Инициализация пользователей и ролей. Это добавление некоторые начальные данные в базу данных во время создания ее структуры или миграции. Это особенно полезно для установки известного состояния данных для тестирования, разработки или для предоставления некоторого контекста для приложения, которое будет построено поверх базы данных.

Создаём класс SeedData, который содержит метод Initialize. Этот метод вызывается при запуске приложения и используется для добавления начальных данных в базу данных. Все это делается с помощью Entity Framework Core.

Внутри метода Initialize, сначала проверяем, есть ли уже какие-либо данные в таблице Doljnosts. Если данные уже существуют, то возвращаем управление, так как база данных уже была инициализирована. Если данных еще нет, то добавляем две новые записи в таблицу Doljnosts. Это делается с помощью метода Add и сохраняется в базу данных с помощью метода SaveChanges.

Создание двух ролей "Administrator" и "Pol" и добавление их в базу данных с помощью метода CreateRole. После создания ролей, создаем двух новых сотрудников и добавляем их в базу данных с помощью метода AddUserToRole. Этот метод также назначает каждому пользователю определенную роль.

Таким образом, при первом запуске приложения, база данных будет автоматически инициализирована с набором начальных данных, что упрощает тестирование и разработку (рис. 2.21-2.22).

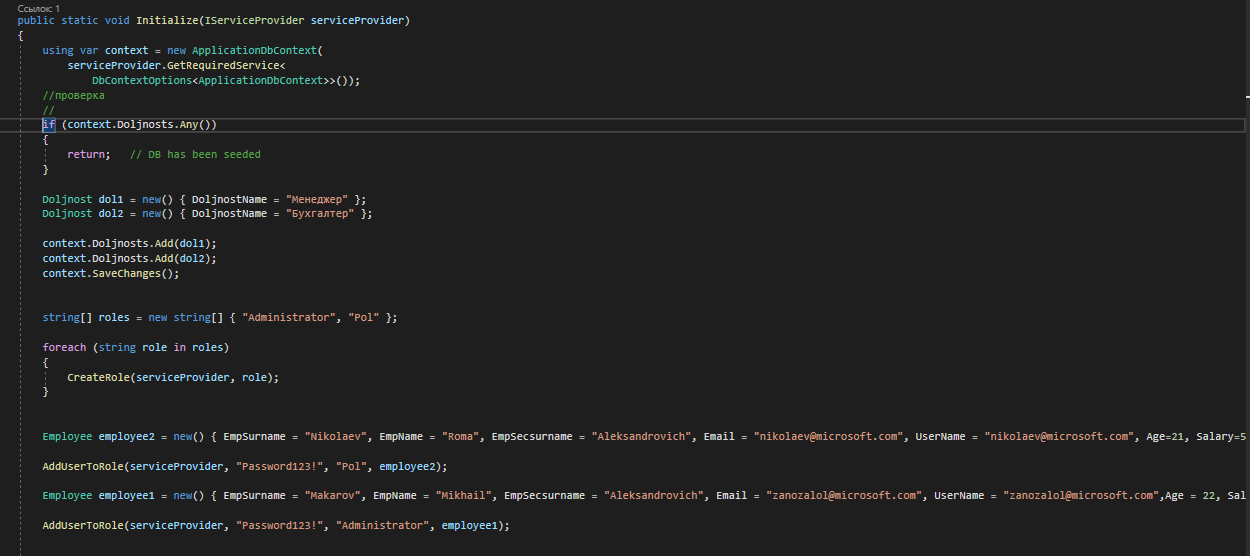


Рисунок 2.21 – Фрагмент экранной копии, проверка и инициализация данных

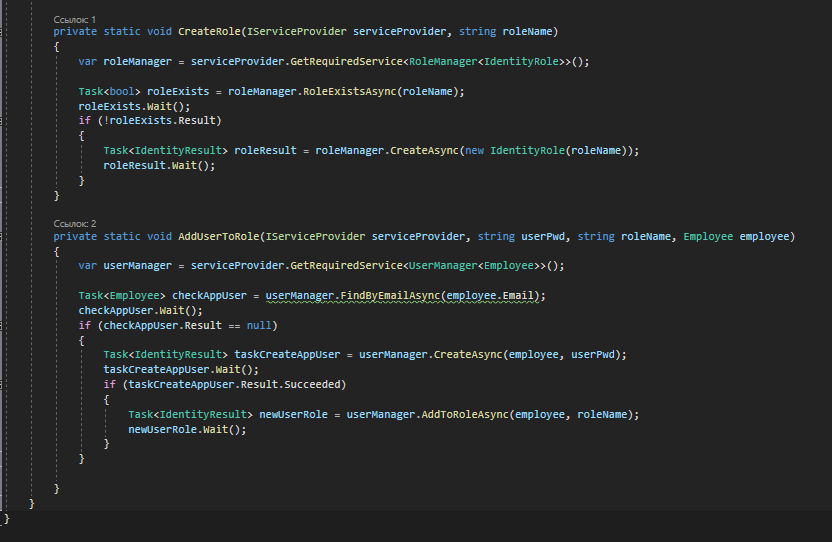


Рисунок 2.22 – Фрагмент экранной копии, создание ролей и пользователей в базе данных

Экспорт и импорт данных из файлов EXCEL. Это важная функция, которая может быть полезна во многих сценариях, таких как обмен данными между различными системами, создание отчетов и обеспечение возможности пользователей загружать свои собственные данные в приложение.

Метод ExportToExcel используется для экспорта данных из таблицы Textiles в файл Excel. Используя библиотеку EPPlus для создания нового файла Excel и заполнения его данными из базы данных. Затем файл отправляется обратно клиенту в виде потока байтов.

Метод ImportFromExcel используется для импорта данных из файла Excel в таблицу Textiles. Считываем файл Excel из входящего запроса, затем разбираем его и добавляем каждую запись в базу данных (рис. 2.23-2.24).

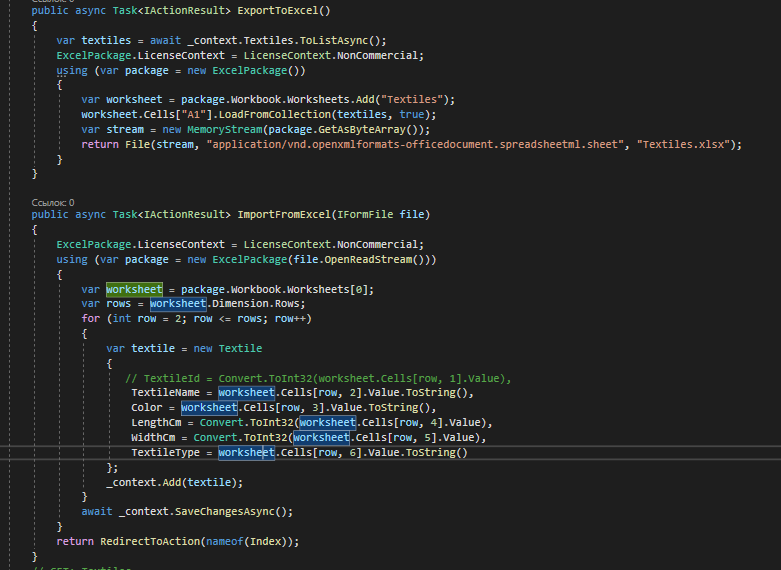


Рисунок 2.23 – Фрагмент экранной копии, методы импорта и экспорта для одной таблицы

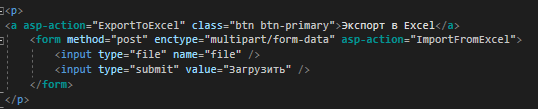


Рисунок 2.24 – Скриншот, показывающий информацию о представлении импорта и экспорта для одной таблицы

После этого реализуем экспорт и импорт данных для трех таблиц. В этом случае используя библиотеку ClosedXML для работы с файлами Excel. Это обеспечивает большую гибкость при работе с данными Excel и позволяет легко управлять стилями, форматированием и другими аспектами файла Excel (рис. 2.25 – 2.26).

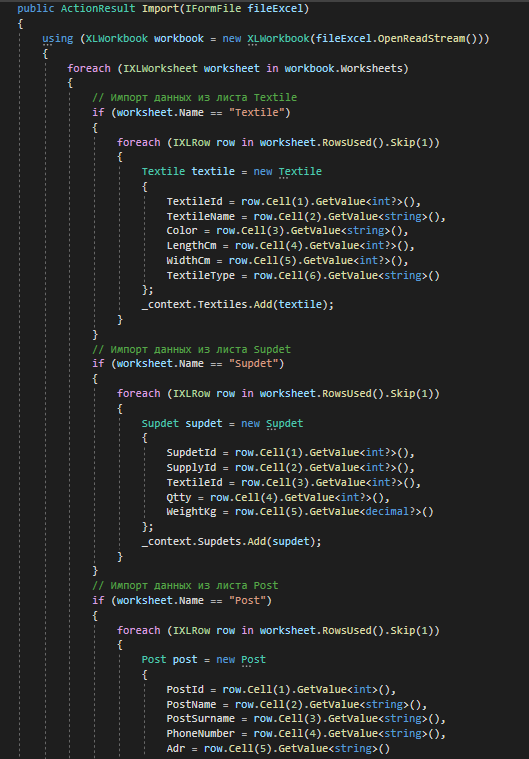


Рисунок 2.25 – Скриншот, показывающий информацию о методах импорта и экспорта для трёх таблиц

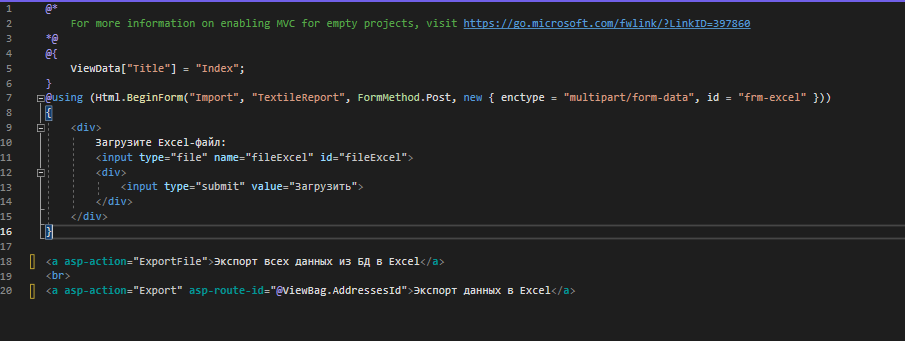


Рисунок 2.26 – Скриншот, показывающий информацию о представлении импорта и экспорта для трёх таблиц

Создание сраницы и таблицы с помощью Blazor. Blazor - это фреймворк для создания интерактивных клиентских веб-приложений с использованием .NET. Создаем контроллер TextileProdController, который использует ApplicationDbContext для взаимодействия с базой данных. В этом контроллере определяем два метода: GetProds и GetProd. Метод GetProds возвращает список всех продуктов, а метод GetProd возвращает конкретный продукт по его идентификатору (рис. 2.27).

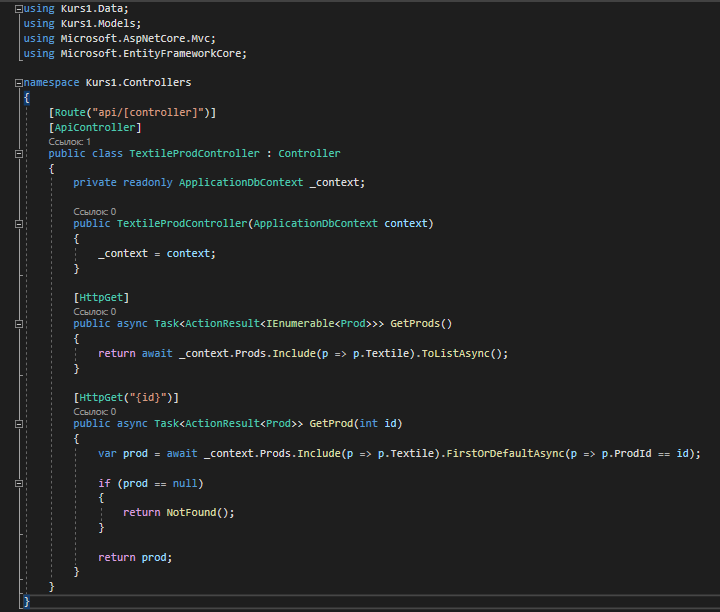


Рисунок 2.27 – Скриншот, показывающий информацию о созданом контроллере

Создаем представление для отображения продуктов. В этом представлении используем цикл foreach для отображения каждого продукта в списке. Для каждого продукта отображается его имя, спецификация, цена, требования к уходу и информация о текстиле(рис. 2.28).

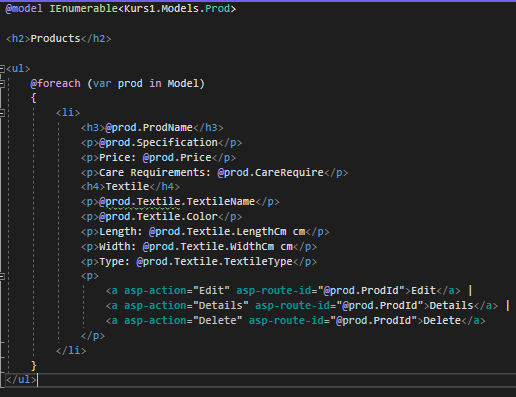


Рисунок 2.28 – Скриншот, показывающий информацию о представлении котороллера

Создаём страницу Blazor, которая использует IHttpClientFactory для получения данных из контроллера и отображения их на странице (рис. 2.29).

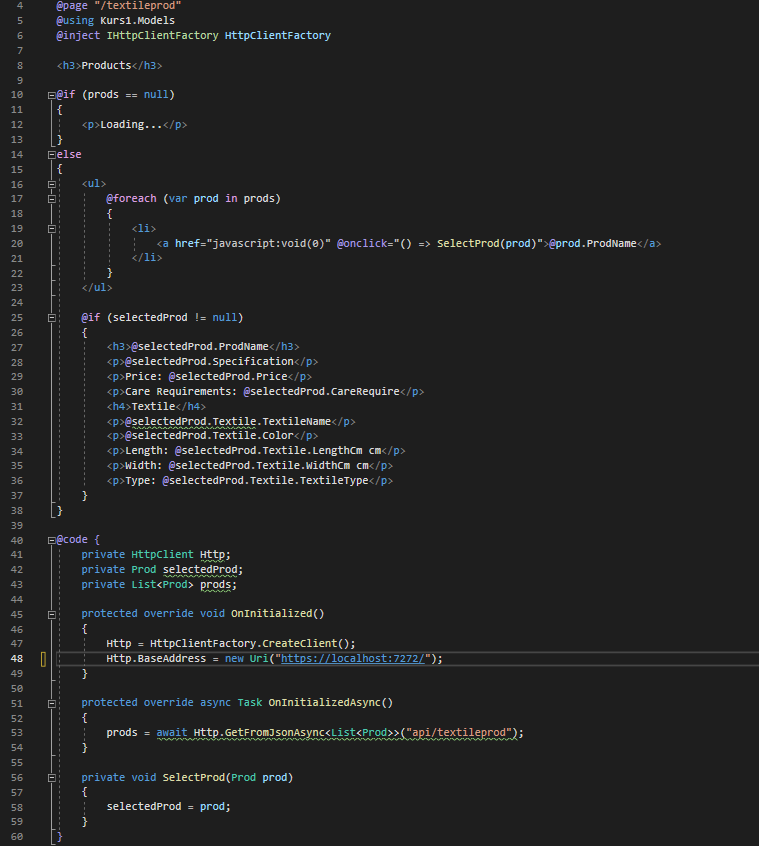


Рисунок 2.29 – Скриншот, показывающий информацию о странице Blazor

Файл Program.cs является точкой входа в приложение и содержит настройку и конфигурацию приложения. Создаёться экземпляр WebApplication с использованием WebApplication.CreateBuilder(args). Этот метод создает и возвращает экземпляр WebApplicationBuilder, который предоставляет API для конфигурации приложения.

Получаем строку подключения к базе данных из конфигурации и используем ее для настройки ApplicationDbContext с помощью UseSqlServer. Также добавляем фильтр исключений разработчика базы данных и настраиваем авторизацию с помощью определенных политик. Настраиваем Identity с использованием AddDefaultIdentity и добавляем роли IdentityRole. Добавляем поддержку контроллеров с представлениями и конфигурацию для Blazor Server. Также настраиваем HTTP-клиент и добавляете поддержку контроллеров с использованием AddNewtonsoftJson для обработки циклических ссылок в JSON. Строим приложение и создам область действия для инициализации данных. Затем настраиваем конвейер обработки HTTP-запросов. Используем обработчики исключений. Настраиваем HTTPS-переадресацию, статические файлы, маршрутизацию и авторизацию (рис. 2.30).

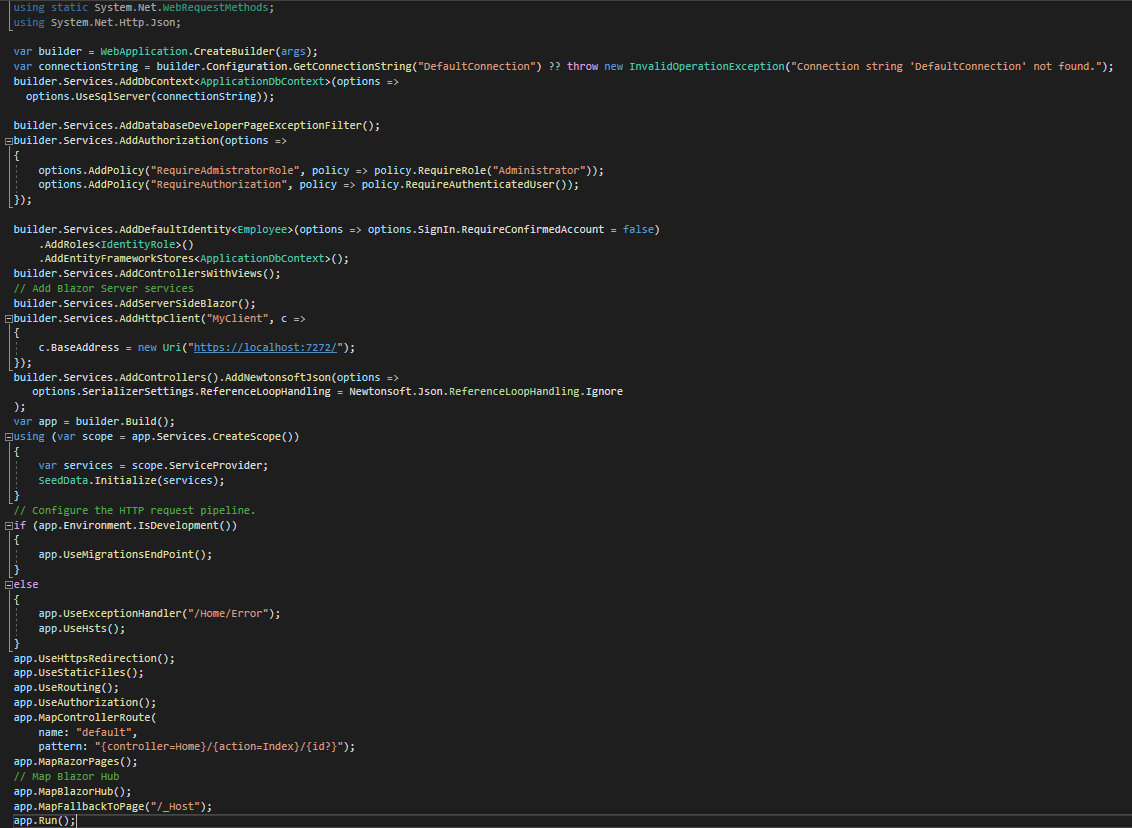


Рисунок 2.30 – Фрагмент экранной копии, файла Program.cs

Проверка всех созданых страниц и заполнение данными (рис. 2.31 – 2.39).

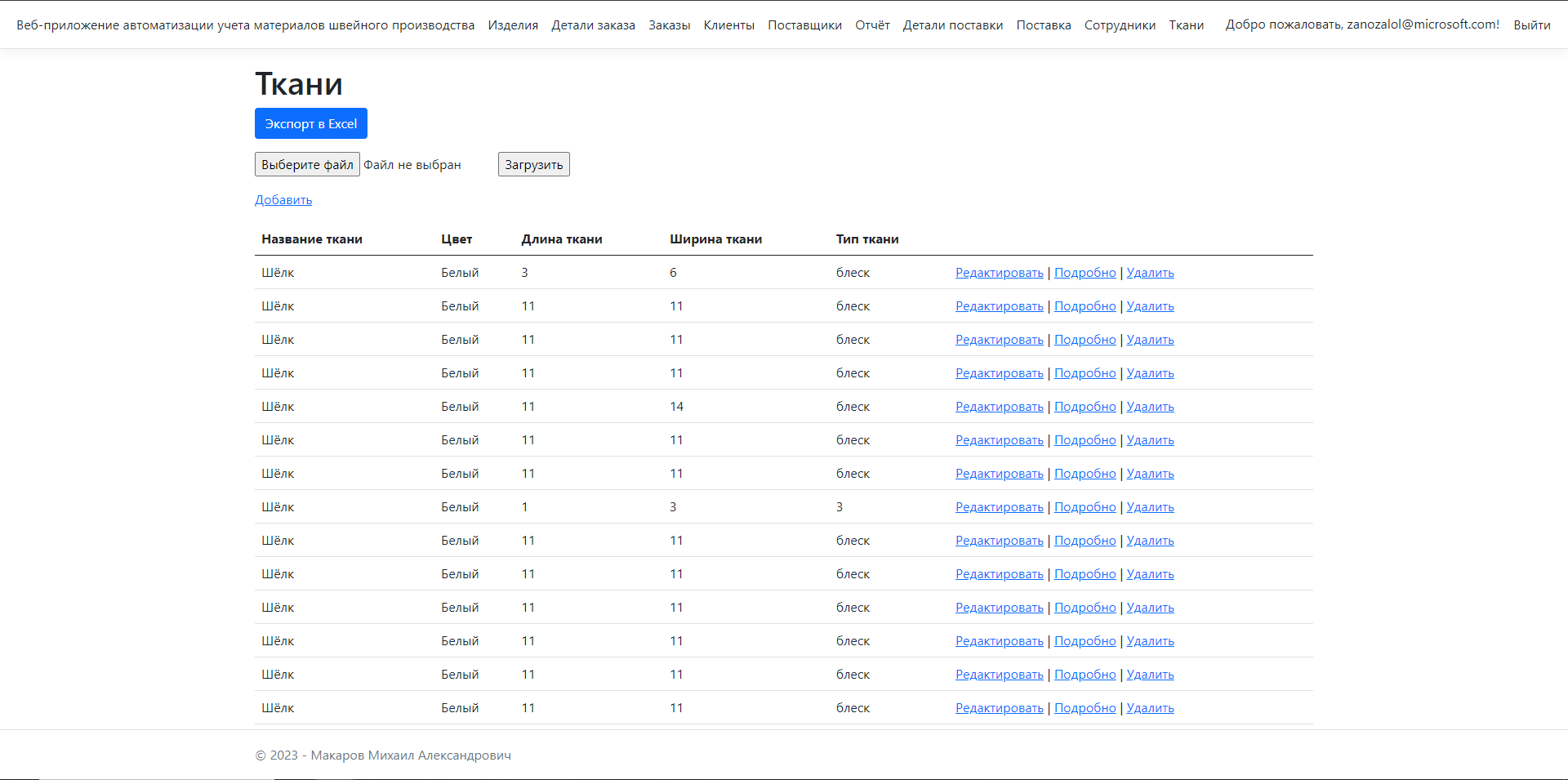


Рисунок 2.31 – Фрагмент экранной копии, ткани

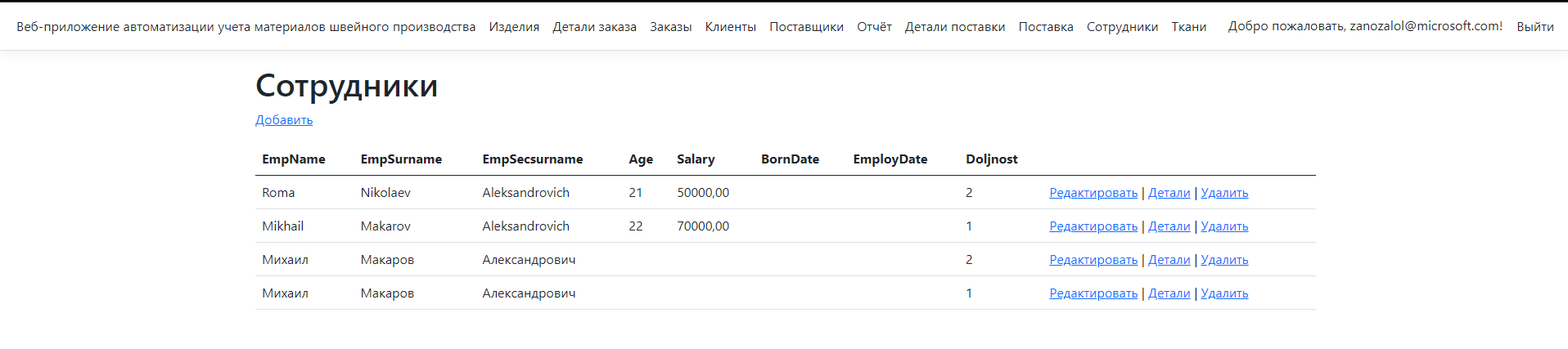


Рисунок 2.32 – Фрагмент экранной копии, сотрудники

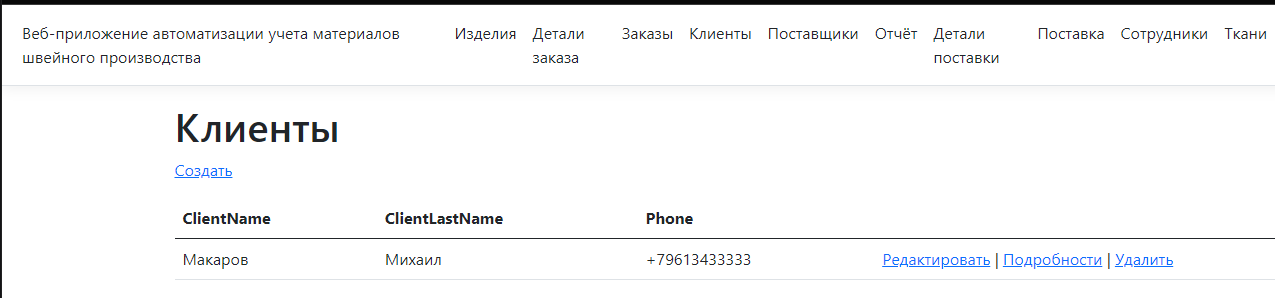


Рисунок 2.33 – Фрагмент экранной копии, клиенты

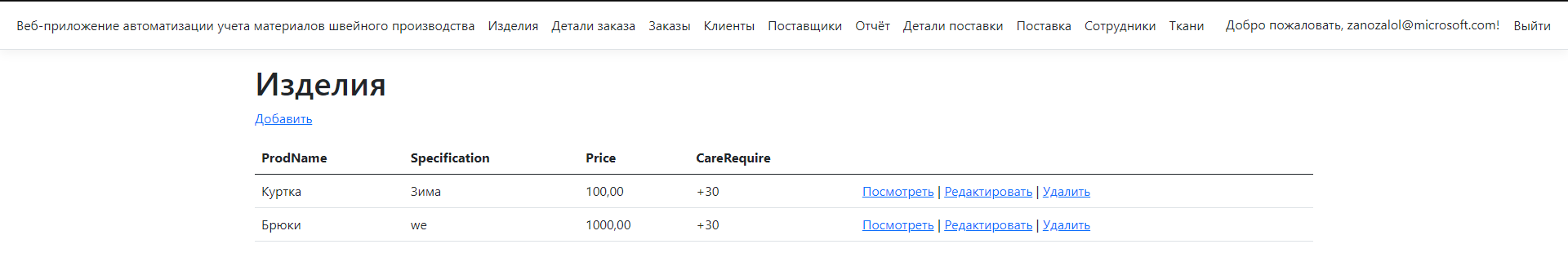


Рисунок 2.34 – Фрагмент экранной копии, изделия

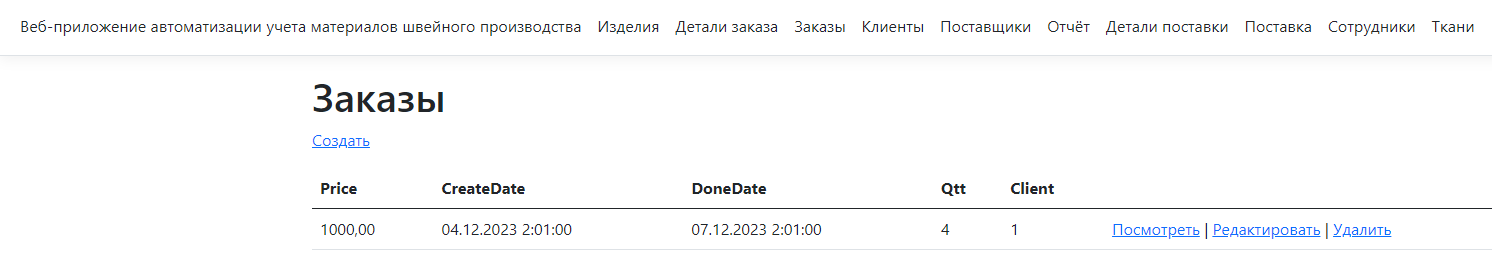


Рисунок 2.35 – Фрагмент экранной копии, заказы

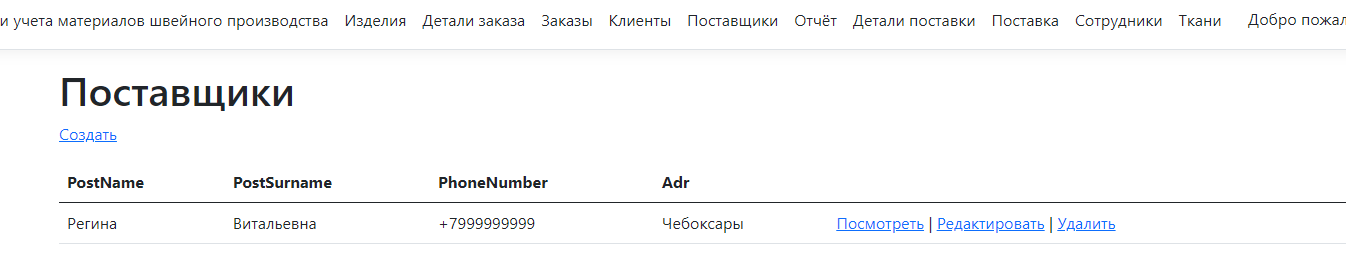


Рисунок 2.36 – Фрагмент экранной копии, поставщики

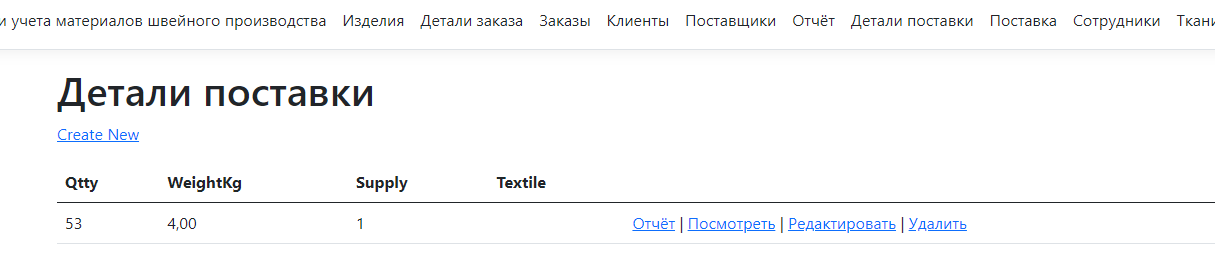


Рисунок 2.37 – Фрагмент экранной копии, детали поставки

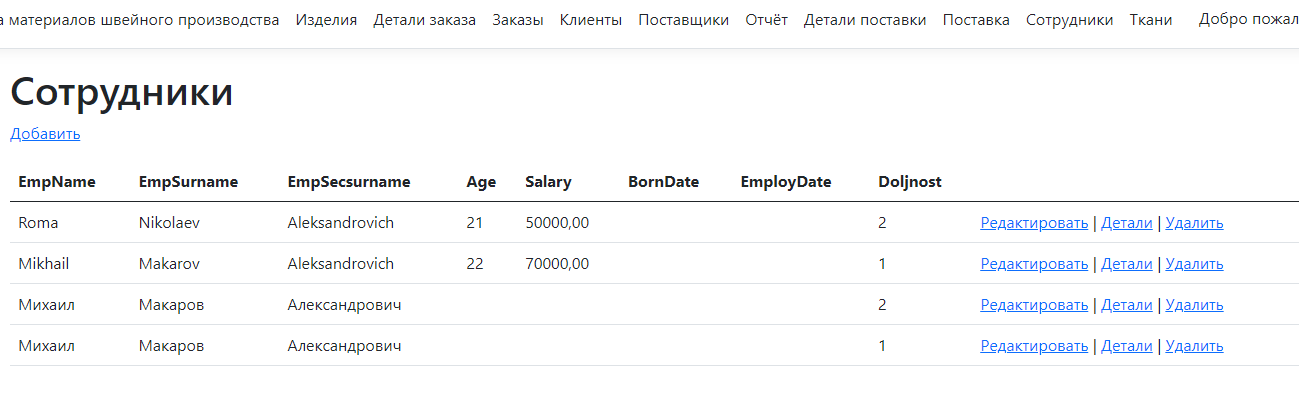


Рисунок 2.38 – Фрагмент экранной копии, сотрудники

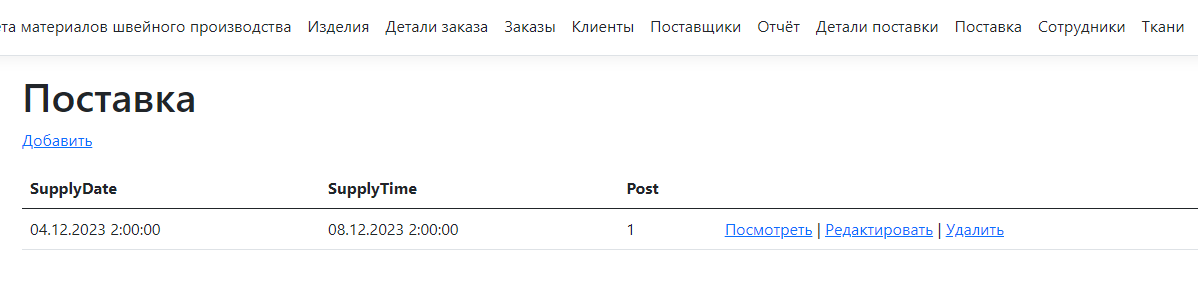


Рисунок 2.39 – Фрагмент экранной копии, поставка

Отдельная страница с импортом и экспортом данных (рис. 40).

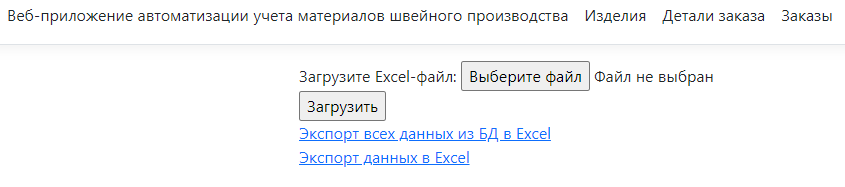


Рисунок 2.40 – Фрагмент экранной копии страницы экспорта и импорта

Страница “Отчёт” созданная с использованием Blazer (рис. 41).

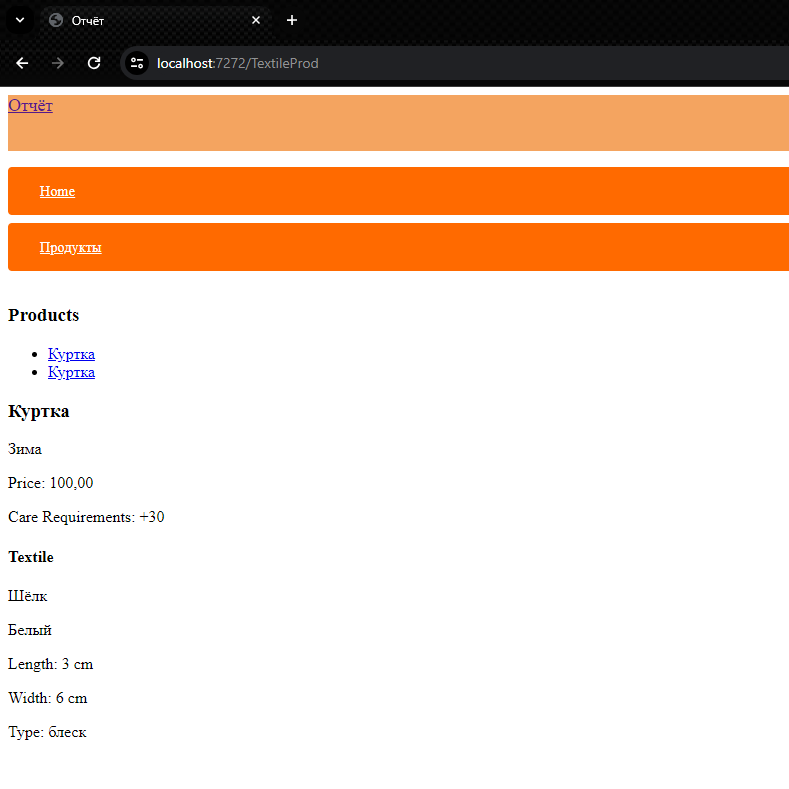


Рисунок 2.41 – Фрагмент экранной копии страницы “Отчёт”

Выводы

Создано веб-приложение, в котором предусмотрены многие функции, позволяющие пользователю эффективно взаимодействовать с базой данных: добавление, редактирование, удаление и хранение информации о клиентах, продукции, заказах клиентов, составе заказа клиентов, сотрудниках, поставках, деталях поставки и вспомогательные функции, такие как экспорт и импорт данных, отчёт по нескольким страницам. Проведено заполнение страниц данными, а также был реализован триггер, производящий проверку данных.

# **Заключение**

Для курсового проекта в качестве предметной области выбрана деятельность швейного производства с целью проектирования для неё веб-приложение, которое будет удобно и практично для работы в данной сфере и которое облегчит оборот документов.

Во время выполнения курсового проекта решены следующие задачи:

* На этапе проведения анализа предметной области рассмотрены два программных продукта:
* сайт «АТЕЛЬЕ ГАРДЕРОБ»;
* ПО "АСТЭР: Управление швейным производством" (Швейка 8).

С их помощью определен перечень функций, которые должны быть реализованы в веб-приложении автоматизации учета материалов швейного производства.

* + - На этапе концептуального проектирования выделены основные действующие лица швейного производства: сотрудник, менеджер, клиент. Определены их функции и составлена диаграмма вариантов использования, а также составлен список объектов, данные которых нужно хранить в системе.
    - На этапе логического проектирования выделены основные сущности. Исходя из них, построены ER-диаграммы. Из разработанных ER-диаграмм, согласно правилам, сформирован предварительный набор отношений для хранения данных (Сотрудник, Заказ, Изделие, СоставЗаказа, Ткань, Клиент, Поставщик, Поставка, СоставПоставки.). А после добавлены неключевые атрибуты в сформированные отношения.
    - На этапе физического проектирования, были выделены основные таблицы, составлена диаграмма базы данных и диаграмма бизнес-процессов idef0.

При проектировании веб-приложения предусмотрены многие функции, позволяющие пользователю эффективно взаимодействовать с ним: добавление, редактирование, удаление и хранение информации о клиентах, продукции, заказах клиентов, составе заказа клиентов, сотрудниках, поставках, деталях поставки и вспомогательные функции, такие как экспорт и импорт данных, отчёт по нескольким страницам.

Цель курсового проекта достигнута, все поставленные задачи выполнены.

# **Список используемых источников**

1. Сайт химчистки «АТЕЛЬЕ ГАРДЕРОБ» [Электронный ресурс] – Режим доступа: https://g-atelier.ru/, свободный. Дата обращения: 12.11.2023
2. Сайт «ИНТУИТ» [Электронный ресурс] – Режим доступа: https://intuit.ru/studies/courses/574/430/lecture/9749, свободный. Дата обращения: 10.11.2023
3. Швейка 8 «АСТЭР: Управление швейным производством» [Электронный ресурс] – Режим доступа: https://v8.1c.ru/small.biz/, свободный. Дата обращения: 12.11.2023
4. Сайт «Майкрософт» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/sql/ssms/, свободный. Дата обращения: 11.11.2023
5. Тейлор А. Г. SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, 2015. - 416 c.
6. Лок Э. ASP.NET CORE в действии - М.: ДМК Пресс, 2021. - 906 c. Дата обращения: 15.11.2023
7. Сайт «Википедия» [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Проектирование\_баз\_данных/, свободный. Дата обращения: 11.11.2023

# 