МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных для сервиса по доставке еды с использованием технологии настройки системы безопасности сервера»

Выполнил студент Плюто Эрик Валерьевич

(Ф.И.О.)

Руководитель работы асс. Нистюк Ольга Александровна

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

И.о. зав. кафедрой ст. преп. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc153773328)

[Постановка задачи 6](#_Toc153773329)

[1 Анализ требований к программному средству 7](#_Toc153773330)

[1.1 Аналитический обзор аналогов 7](#_Toc153773331)

[1.2 Определение требований 9](#_Toc153773332)

[2 Проектирование базы данных 11](#_Toc153773333)

[2.1 Разработка модели базы данных 11](#_Toc153773334)

[2.2 Вывод 14](#_Toc153773335)

[3 Разработка объектов базы данных 15](#_Toc153773336)

[3.1 Таблицы 15](#_Toc153773337)

[3.2 Процедуры 15](#_Toc153773338)

[3.3 Функции 15](#_Toc153773339)

[3.4 Триггеры 16](#_Toc153773340)

[4 Описание процедур импорта и экспорта 17](#_Toc153773341)

[5 Тестирование производительности 18](#_Toc153773342)

[5.1 Вывод по разделу 18](#_Toc153773343)

[6 Описание технологий и их применение в базе данных 19](#_Toc153773344)

[6.1 Резервное копирования и восстановление 19](#_Toc153773345)

[6.2 Настройка аудита 20](#_Toc153773346)

[6.3 Вывод по разделу 21](#_Toc153773347)

[7 Краткое описание приложения для демонстрации 22](#_Toc153773348)

[8 Руководство пользователя 23](#_Toc153773349)

[Заключение 31](#_Toc153773350)

[Список литературы 32](#_Toc153773351)

[ПРИЛОЖЕНИЕ A 33](#_Toc153773352)

[ПРИЛОЖЕНИЕ Б 35](#_Toc153773353)

[ПРИЛОЖЕНИЕ В 37](#_Toc153773354)

[ПРИЛОЖЕНИЕ Г 38](#_Toc153773355)

ВВЕДЕНИЕ

Система управления базами данных (СУБД) представляет собой программное обеспечение, которое используется для создания, управления и обработки баз данных.

СУБД позволяет пользователям определить структуру базы данных, создавать таблицы, добавлять, изменять и удалять данные, а также выполнять запросы для извлечения и модификации информации. Она обеспечивает целостность данных, контроль доступа и защиту от несанкционированного доступа.

Современные СУБД поддерживают различные модели данных, такие как реляционная модель, объектно-ориентированная модель, иерархическая модель.

Реляционная база данных – база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Microsoft SQL Server.

СУБД используются во многих сферах: от банков и страховых компаний до интернет-сервисов и мобильных приложений. Они являются неотъемлемой частью современной информационной технологии и позволяют эффективно управлять большими объемами данных, a также предоставляют мощные инструменты для оптимизации запросов, индексации и резервного копирования данных.

СУБД играют ключевую роль в организации и обработке данных, обеспечивая их целостность, доступность и безопасность. Они позволяют эффективно управлять информацией и представляют важную технологию для работы с базами данных.

Также требуется разработать приложение для демонстрации функциональности базы данных и взаимодействия с ней. Это приложение было реализовано на платформе ASP.NET Core, который является фреймворком для создания веб-приложений на платформе .NET.

Постановка задачи

Целью данной курсовой работы является создание базы данных для веб-приложения, предназначенного для поиска товаров от разных ресторанов и оформления заказов. Также было необходимо настроить систему безопасности СУБД.

Задачи курсовой работы: определить необходимые сущности, создать на основе сущностей таблицы и установить связи между ними, инкапсулировать внутреннюю логику базы данных от пользователей, добиться приемлемой скорости работы базы данных, реализовать экспорт и импорт данных в определенном формате.

База данных будет обладать следующими функционалом:

* реализация ролей администратора, администратора компании и пользователя;
* управление пользователями (регистрация, авторизация, редактирование, удаление);
* управление компаниями (добавление, редактирование, удаление);
* управление категориями (добавление, редактирование, удаление);
* управление товарами (добавление, редактирование, удаление);
* управление заказами (добавление, изменение статуса, удаление);
* поиск товаров (по названию, по аллергену) и компаний (по названию).

1. Анализ требований к программному средству

1.1 Аналитический обзор аналогов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

«Яндекс Еда» – это один из самых популярных сервисов по доставке еды в СНГ. Данный сервис предоставляет клиентам возможность выбрать и заказать еду из широкого ассортимента ресторанов и кафе. Этот сервис успешно справляется с поставленными задачами и имеет определенные преимущества.

Преимущества Яндекс Еды:

* компания имеет широкий выбор блюд, которые разбиты на различные категории;
* система поиска и фильтрации помогает быстро найти нужное блюдо;
* Яндекс Еда имеет удобную систему отслеживания заказа, что позволяет клиентам точно знать, когда будет доставлено их блюдо;
* компания имеет гибкую систему скидок и акций, что делает ее услуги более доступными для потребителей.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

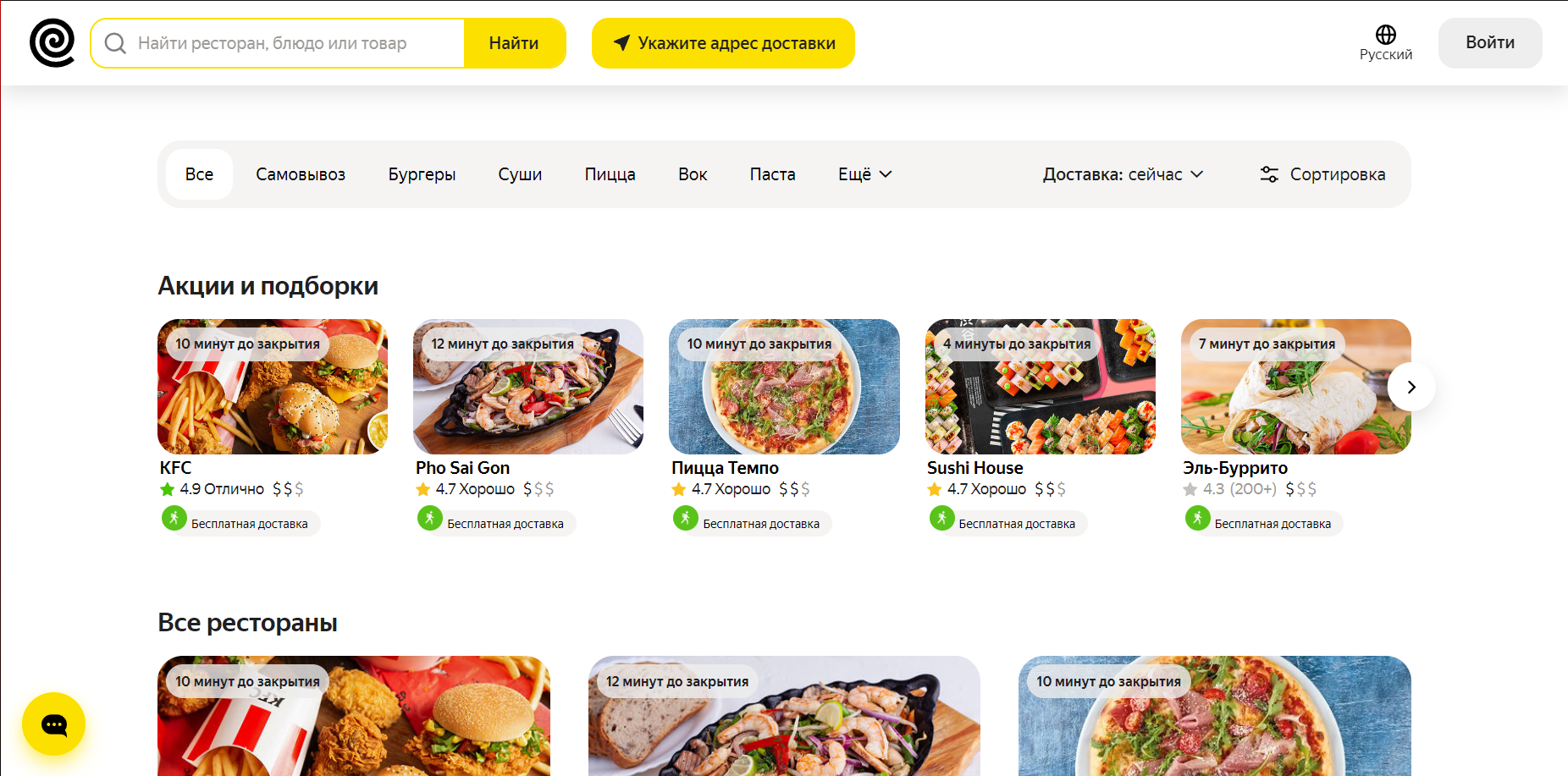


Рисунок 1.1 – Интернет-ресурс «Яндекс Еда»

В целом, Яндекс Еда является удобным и популярным сервисом, обеспечивающим широкий выбор блюд, быструю и удобную доставку, простоту использования и доступные цены.

Еще одним альтернативным решением, которое может использоваться для заказа еды на дом, является интернет-ресурс «Easysoup». На сайте можно выбрать блюдо из широкого ассортимента супов, салатов и десертов, а также указать необходимое количество порций и адрес доставки. Это позволяет пользователям быстро и удобно заказать еду.

Сайт Easysoup имеет удобный функционал корзины, который позволяет пользователям добавлять и удалять товары, а также изменять количество порций. Кроме того, корзина автоматически рассчитывает стоимость заказа, что позволяет пользователям контролировать свой бюджет.

На сайте Easysoup есть подробная информация о каждом блюде, включая описание, ингредиенты, калорийность и цену. Это позволяет пользователям выбирать блюда, исходя из своих предпочтений потребностей.

Интерфейс интернет-ресурса представлен на рисунке 1.2.

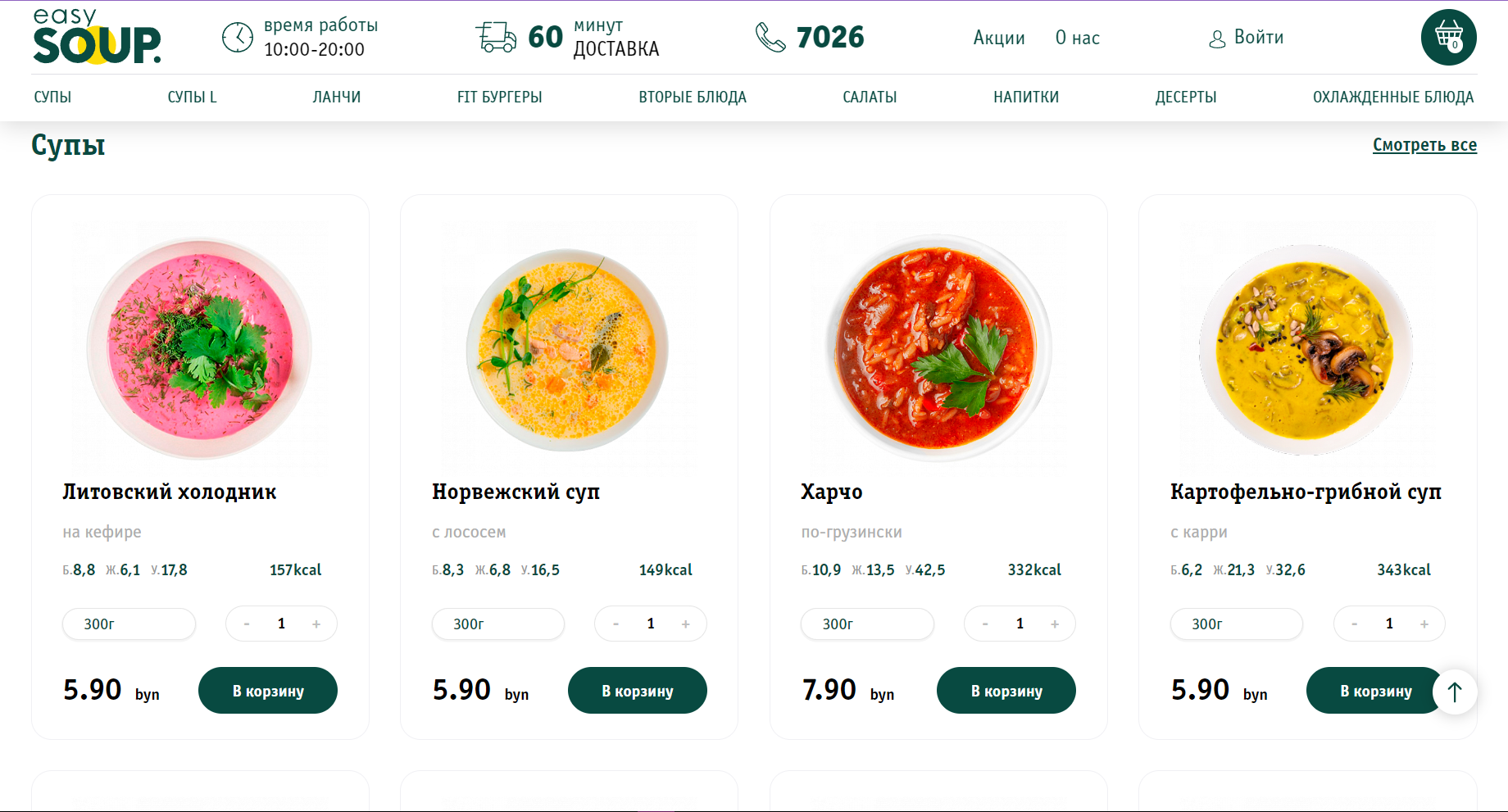


Рисунок 1.2 – Интернет-ресурс «Easysoup»

В целом, сайт Easysoup имеет широкий функционал, который позволяет пользователям быстро и удобно заказывать еду на дом или в офис. Сайт имеет интуитивно понятный интерфейс, удобный функционал корзины и фильтры поиска, а также предоставляет подробную информацию о каждом блюде. Кроме того, сайт Easysoup регулярно проводит промо-акции и предлагает скидки на заказы, что позволяет пользователям экономить деньги и получать дополнительные бонусы.

Другой интернет-ресурс, который также предлагает услуги доставке еды – это «Васiлькi». Товары, как и у аналогов, разбиты на категории, что помогает быстро находить нужное блюдо.

Сайт Васiлькi предоставляет дополнительную возможность онлайн-бронирования столика в ресторане. Пользователи могут выбрать удобное время и дату, а также указать количество гостей и контактную информацию. Это позволяет пользователям забронировать столик быстро и удобно, без лишних звонков и переговоров.

Из минусов можно отметить устаревший дизайн сайта и отсутствие поиска по товарам.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

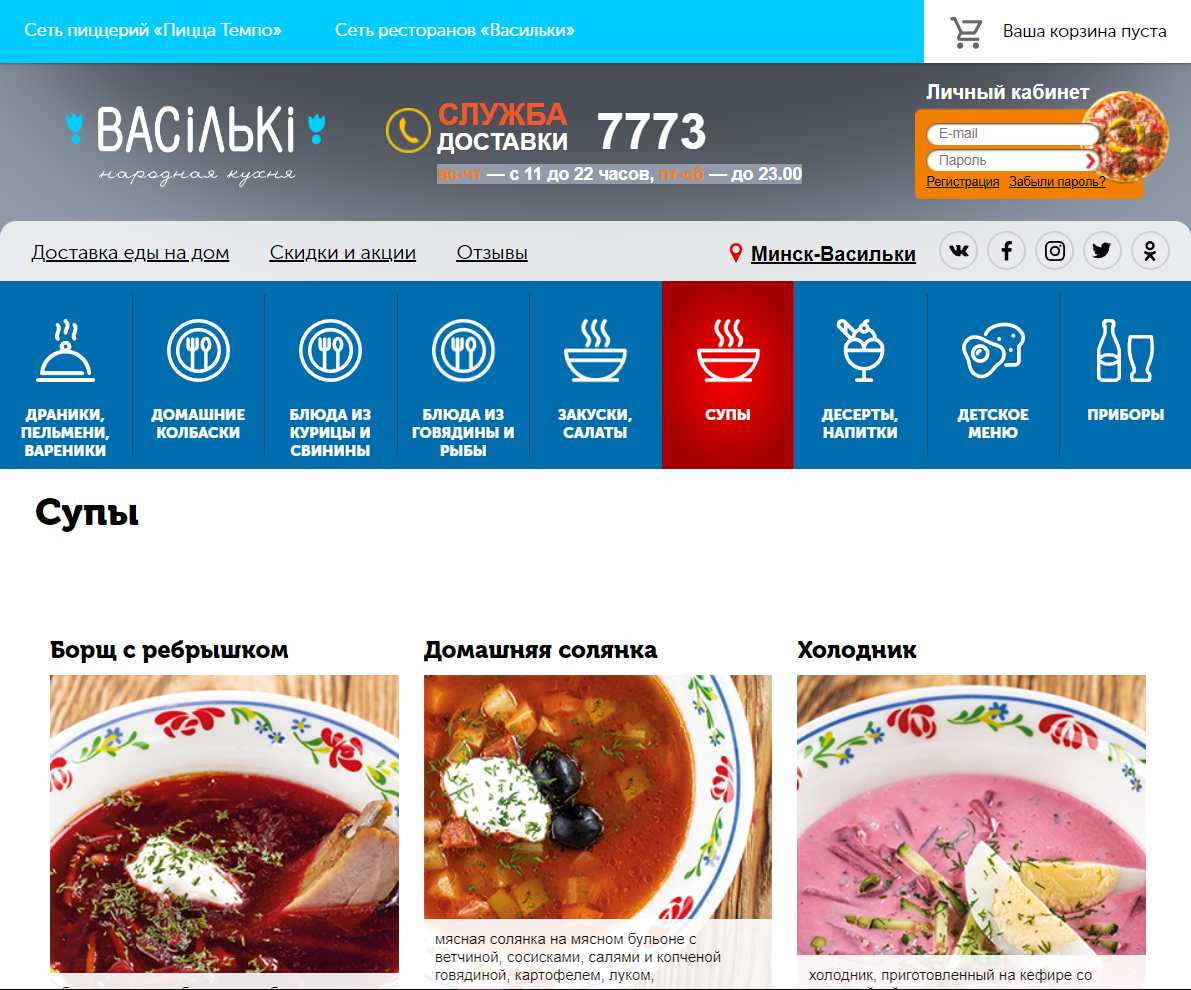


Рисунок 1.3 – Интернет-ресурс «Васiлькi»

Сайт Васiлькi имеет широкий функционал, который позволяет пользователям быстро и удобно заказывать еду на дом или в офис, а также бронировать столики в ресторане.

1.2 Определение требований

Обзор вышеперечисленных известных аналогов решений позволяет проанализировать все преимущества и недостатки, сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Оно должно обеспечивать возможность выполнения перечисленных ниже функций:

* управление базой данных (добавление, изменение и удаление компаний, просмотр всех заказов с возможностью очистки, просмотр истории заказов со стороны администратора, добавление, изменение и удаление категорий и товаров, просмотр и изменение статуса заказов, просмотр истории заказов компании со стороны администратора компании);
* возможность пользователя зарегистрироваться или войти в существующую учетную запись;
* возможность добавления товаров в корзину и оформление пользователем заказа;
* возможность поиска компаний и товаров;
* возможность редактирования и удаления аккаунта;
* возможность просмотра дополнительной информации о товаре при нажатии на него;
* возможность просмотра всех аккаунтов и изменение их привилегий администратором.

Важно установить роли пользователей и определить их сценарии использования системы. Эти сценарии описывают, как каждый пользователь будет взаимодействовать с системой в зависимости от своей роли. Это помогает определить, какие функции будут доступны для каждой роли, какие данные будут доступны, а также как будет организована навигация в системе. Обычно эти сценарии представляются в виде диаграмм UML, которые визуализируют взаимодействие между пользователями и системой.

Роли пользователей представляют собой определенный набор прав, предоставляемых пользователям в системе. В зависимости от роли каждый пользователь может получить доступ к различным функциям системы. В данном проекте роли пользователей включают в себя: Admin, Company Admin, User.

На основе предоставленного списка ролей необходимо построить диаграмму вариантов использования. Диаграмма вариантов использования изображена на рисунке 1.4.

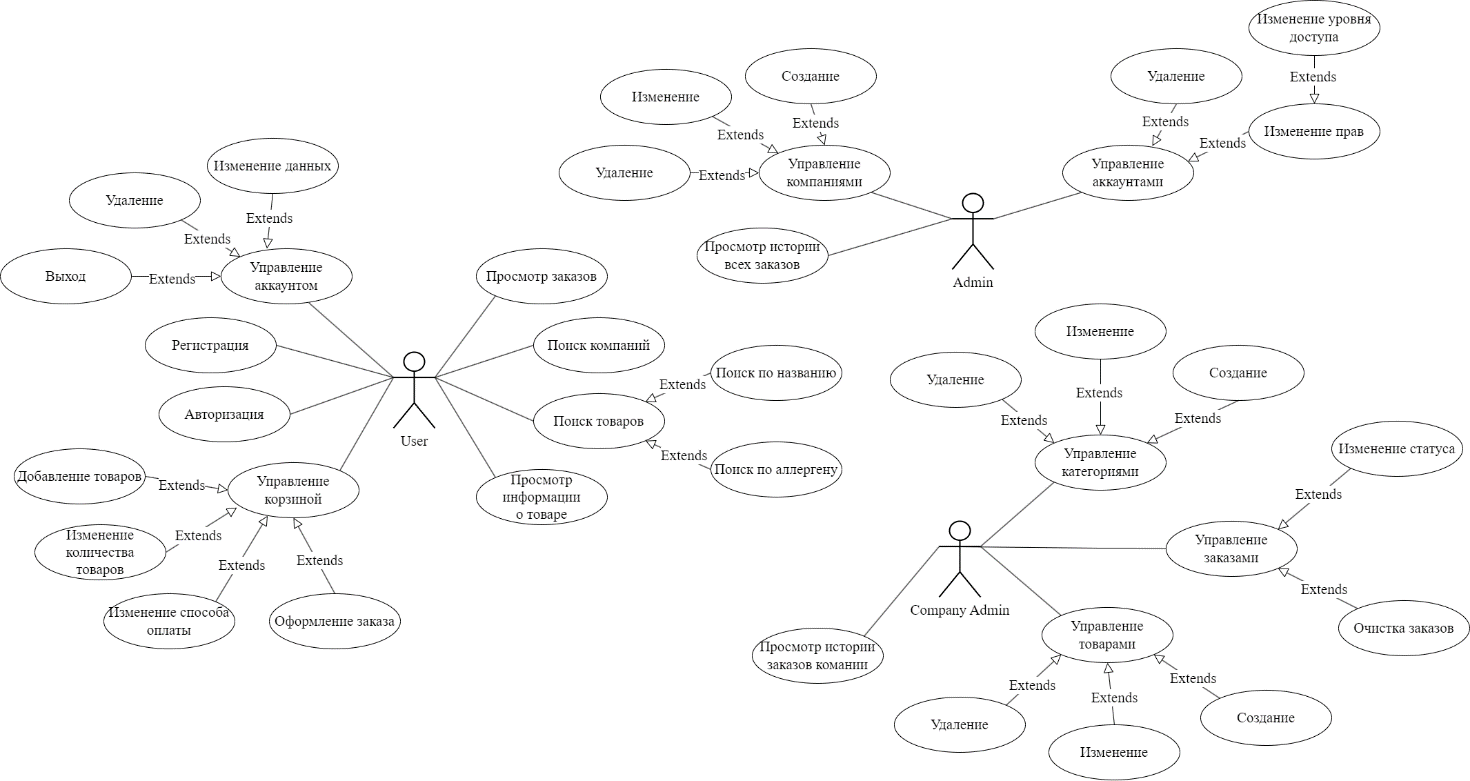


Рисунок 1.4 − Диаграмма вариантов использования

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

2 Проектирование базы данных

2.1 Разработка модели базы данных

Одним из ключевых моментов при проектировании и создании базы данных является грамотный анализ предметной области базы данных. Далее необходимо составить такую модель данных, которая будет правильно отражать то, как с этими с данными в общем, и этой моделью, в частности, подразумевается взаимодействовать.

Результатом корректно проведённого анализа, проектирования, и разработки, является модель, способная предоставить функционально все необходимые возможности для конкретного типа пользователя.

В рамках этого этапа была создана модель, которая включает 7 сущностей:

* аллерген;
* категория;
* компания;
* история;
* заказ;
* продукт;
* пользователь.

На рисунке 2.1 отображена модель базы данных, полученная в ходе анализа предметной области и необходимого функционала. Между сущностями, представленными таблицами, также настроены связи, отражающие их взаимосвязь.

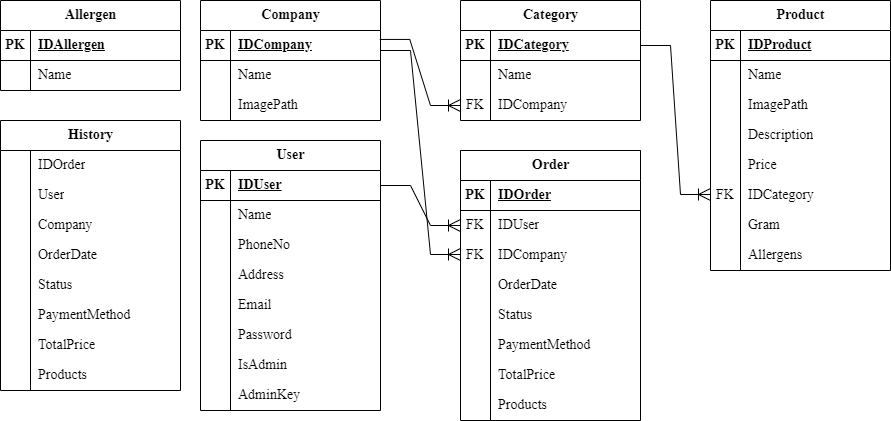


Рисунок 2.1 – Схема базы данных

В структуру базы данных входят следующие таблицы: Allergen, History, Company, Category, Product, User, Order. Далее представлено описание каждой из них.

Таблица Allergen используется для хранения данных об аллергенах. Её структура представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы Allergen

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDAllergen | Int, primary key | Идентификатор аллергена |
| Name | Nvarchar(50), not null | Название аллергена |

Таблица History используется для хранения данных об изменении и создании заказов, которые представляют собой номер заказа, адрес электронной почты заказчика, название ресторана, в котором был оформлен заказ, дату создания заказа или дату изменения его состояния, состояние заказа («Принят», «В пути», «Доставлен», «Отменён»), метод оплаты («Наличные», «Карта»), итоговую стоимость заказа и список заказанных товаров. Её структура представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы History

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDOrder | Int, not null | Номер заказа |
| User | Nvarchar(50), not null | Адрес электронной почты заказчика |
| Company | Nvarchar(50), not null | Название ресторана |
| OrderDate | Datetime, not null | Дата изменения или создания заказа |
| Status | Nvarchar(50), not null | Состояние заказа |
| PaymentMethod | Nvarchar(50), not null | Метод оплаты |
| TotalPrice | Numeric(8, 2), not null | Итоговая стоимость заказа |
| Products | Nvarchar(300) | Список заказанных товаров |

Таблица Company используется для хранения данных о ресторанах, которые представляют собой идентификатор ресторана, его название и путь к файлу обложки ресторана, который будет отображаться в веб-приложении. Её структура представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы Company

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDCompany | Int, primary key, identity(1, 1) | Идентификатор ресторана |
| Name | Nvarchar(50), not null | Название ресторана |
| ImagePath | Nvarchar(150), not null | Путь к файлу обложки ресторана |

Таблица Category используется для хранения данных о категориях ресторанов, которые представляют собой идентификатор категории, её название и идентификатор компании, к которой принадлежит категория. Её структура представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы Category

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDCategory | Int, primary key, identity(1, 1) | Идентификатор категории |
| Name | Nvarchar(50), not null | Название категории |
| IDCompany | Int, foreign key, not null | Идентификатор компании |

Таблица Product используется для хранения данных о товарах ресторанов, которые представляют собой идентификатор продукта, его название, путь к файлу обложки товара, описание товара, его стоимость, идентификатор категории, к которой принадлежит товар, граммовку товара и список аллергенов, которые содержатся в товаре. Её структура представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы Product

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDProduct | Int, primary key, identity(1, 1) | Идентификатор продукта |
| Name | Nvarchar(30), not null | Название продукта |
| ImagePath | Nvarchar(100), not null | Путь к файлу обложки товара |
| Description | Ntext, not null | Описание |
| Price | Numeric(8, 2), not null | Цена единицы товара |
| IDCategory | Int, foreign key, not null | Идентификатор категории |
| Gram | Int, not null | Граммовка единицы товара |
| Allergens | Nvarchar(50) | Аллергены товара |

Таблица User используется для хранения данных о пользователях, которые представляют собой идентификатор пользователя, его имя, номер мобильного телефона, адрес, электронную почту, пароль, переменную, указывающую права пользователя (0 – пользователь, 1 – администратор), и переменную, отвечающую за уровень этих прав (0 – полный доступ, значение больше 0 – идентификатор компании, иначе – пользователь). Её структура представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы User

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDUser | Int, primary key, identity(1, 1) | Идентификатор пользователя |
| Name | Nvarchar(50), not null | Имя пользователя |
| PhoneNo | Nvarchar(50), not null | Номер мобильного телефона |
| Address | Nvarchar(50), not null | Адрес |
| Email | Nvarchar(50), not null | Адрес электронной почты |
| Password | Varbinary(500), not null | Хешированный пароль |
| IsAdmin | Bit, not null | Права пользователя |
| AdminKey | Int | Уровень прав |

Таблица Order используется для хранения данных заказах, которые представляют собой идентификатор заказа, идентификатор пользователя, идентификатор ресторана, дату создания заказа, состояние заказа («Принят», «В пути», «Доставлен», «Отменён»), метод оплаты («Наличные», «Карта»), итоговую стоимость заказа и список заказанных товаров. Её структура представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы Order

|  |  |  |
| --- | --- | --- |
| Столбец | Тип, ограничение целостности | Описание |
| IDOrder | Int, primary key, identity(1, 1) | Идентификатор заказа |
| IDUser | Int, foreign key, not null | Идентификатор пользователя |
| IDCompany | Int, foreign key, not null | Идентификатор ресторана |
| OrderDate | Datetime, not null | Дата создания заказа |
| Status | Nvarchar(50), not null | Состояние заказа |
| PaymentMethod | Nvarchar(50), not null | Метод оплаты |
| TotalPrice | Numeric(8, 2), not null | Итоговая стоимость заказа |
| Products | Nvarchar(300) | Список заказанных продуктов |

Каждая таблица служит определенной цели, обеспечивая хранение данных о компаниях, категориях, товарах, аллергенах, пользователях и заказах в приложении.

2.2 Вывод

Разработка модели базы данных является важным этапом процесса создания и управления базой данных. В данном разделе были рассмотрены ключевые аспекты этого процесса, включая определение сущностей и их атрибутов, определение связей между сущностями и выбор соответствующей модели данных.

Описания информационных объектов являются важным элементом базы данных, поскольку они обозначают, какие данные будут использоваться в приложении, а также указывают, как эти данные будут храниться, обрабатываться и взаимодействовать друг с другом. В данном проекте осуществлено детальное описание таких объектов, как аллергены, компании, категории, товары, заказы, пользователя и история заказов.

В целом, создание проектирование является важным этапом в разработке приложения, поскольку оно не только определяет необходимые компоненты, но и помогает построить функциональное и эффективное приложение, обеспечивая понимание внутренней структуры и взаимодействия его элементов.

3 Разработка объектов базы данных

3.1 Таблицы

Таблицы баз данных представляют собой структурированные наборы данных, которые хранятся в базе данных. Они состоят из строк (записей) и столбцов (полей), где каждое поле содержит определенный тип данных и хранит конкретное значение.

Основная цель таблиц баз данных – хранить структурированную информацию и обеспечивать возможность эффективного доступа и модификации данных.

В данном проекте базы данных содержится семь таблиц, каждая из которых описана в разделе 2. Соответствующий SQL-код для создания этих таблиц приведен в приложении А.

3.2 Процедуры

Процедуры в базе данных являются набором инструкций или операций, которые выполняют определенные задачи и выполняются внутри базы данных. Они позволяют упростить и стандартизировать выполнение операций над данными, а также повысить эффективность и безопасность работы с базой данных.

Хранимые процедуры могут включать в себя основные операторы для создания, изменения и удаления данных (DDL, DML и TCL), использовать конструкции TRY/CATCH, работать с курсорами, временными таблицами и другими средствами языка T-SQL.

Примеры процедур из курсового проекта, выполняющих различные операции:

* Удаление: DeleteProduct – предназначена для удаления товара из базы данных; DeleteUser – для удаления пользователя из базы данных.
* Изменение: UpdateRights – предназначена для обновления прав пользователя таблицы User; ChangeStatus – для изменения статуса заказа.
* Вставка: CreateCategory – для создания новой категории компании; CreateCompany – для создания нового ресторана в базе данных.

В приложении Б представлен листинг SQL-кода для создания процедур, описанных выше.

3.3 Функции

Функции в базе данных – это особый тип процедур, которые возвращают значения и могут использоваться внутри выражений SQL или в других процедурах. Они выполняют определенные операции и возвращают результат в качестве значения.

Пользовательские функции, по сути, являются программными модулями, которые разработчик определяет и сохраняет в базе данных. Они могут быть написаны на языках SQL, PL/SQL, T-SQL или других языках программирования, поддерживаемых СУБД. Пользовательские функции могут выполнять более сложные операции и могут быть вызваны из других запросов, процедур или функций.

Пример функций из курсового проекта выполняющие различные операции:

* CheckEmailFunc: эта функция предназначена для проверки адреса электронной почты на дублирование в базе данных. Принимает параметр email, адрес электронной почты, который нужно проверить. Возвращаемый тип данных: bit.
* CompanyAveragePrice: эта функция возвращает среднюю стоимость товаров ресторана. Принимает параметр CompanyId, идентификатор ресторана. Возвращаемый тип данных: numeric(8, 2).
* AveragePrices: эта функция предназначена для вычисления средних стоимостей товаров всех ресторанов базы данных. Не принимает параметров. Возвращаемый тип данных: table.

В приложении В представлен листинг SQL-кода для создания функций, осуществляющих различные операции.

3.4 Триггеры

Триггеры в базе данных – это специальные типы процедур, которые автоматически выполняются при возникновении определенных событий или условий в базе данных. Они могут быть определены для таблицы или базы данных и могут реагировать на операции вставки, обновления или удаления данных.

Триггеры могут быть использованы для выполнения определенных операций или проверок данных в ответ на определенные события. Например, при вставке новой строки в таблицу, триггер может запускать процедуру, которая автоматически обновляет другие таблицы или выполняет дополнительную проверку данных.

Для базы данных был создан триггер history\_trigger для таблицы Order. Этот триггер был создан чтобы отслеживать создание и изменение заказов. Триггер выполняется после операций вставки и обновления (AFTER INSERT, UPDATE) и вставляет данные о заказе в таблицу History.

Листинг создания триггера представлен в приложении Г.

4 Описание процедур импорта и экспорта

База данных имеет возможность экспортировать и импортировать данные для таблицы User в формате JSON. Это может быть полезно в случае необходимости переноса данных на другой сервер или резервного копирования данных.

Для экспорта данных в формате JSON была создана процедура UserToJson. Процедура представлена на листинге 4.1.

|  |
| --- |
| CREATE procedure [dbo].[UserToJson]  as  begin try  DECLARE @cmd varchar(8000), @out varchar(8000)  SET @out = (select \* from [dbo].[User] for json path)  print @out  SET @cmd = 'echo ' + @out + ' > C:\Users\Erik\Desktop\3course\Delivery\Json\User.json'  print @cmd  EXEC delivery..xp\_cmdshell @cmd  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH |

Листинг 4.1 – Процедура UserToJson

Для импорта данных в формате JSON была создана процедура UserFromJson.

Таким образом, пользователи базы данных могут легко экспортировать и импортировать данные в форматах JSON, что делает управление базой данных более удобным и эффективным.

5 Тестирование производительности

Производительность базы данных – это способность базы данных обрабатывать запросы и транзакции быстро и эффективно. Высокая производительность базы данных может быть достигнута с помощью эффективного использования индексов, хорошей организации структуры базы данных и оптимизации запросов. Хорошая производительность базы данных важна для обеспечения эффективной работы системы и удовлетворения потребностей пользователей.

Для тестирования производительности в таблицу Product было добавлено 100000 строк. Для данной задачи был создан анонимный блок, который с помощью ранее написанной процедуры CreateProduct вставляется данные в таблицу.

Разработанный анонимный блок представлен на листинге 5.1.

DECLARE @i int = 1;

WHILE @i <= 100000

BEGIN

exec CreateProduct @i, 'test', 'description', 5, 3005, 10, '1';

set @i = @i + 1;

END;

Листинг 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Был выполнен select-запрос на поиск элемента по имени к таблице Product без использования индекса, был просмотрен план данного запроса, его стоимость составляла 0,79.

Для улучшения производительности был создан индекс на таблицу Product, а именно индекс к полю Name. После этого стоимость select-запроса стала равна 0,007.

Соответственно мы можем сделать вывод, что индексы могут улучшить производительность базы данных за счет ускорения поиска и обработки данных. Они создаются для определенных столбцов и позволяют ускорить поиск строк в таблице по этим столбцам. Индексы также могут ускорить сортировку и группировку данных

5.1 Вывод по разделу

В данном разделе было проведено нагрузочное тестирование для проверки наличия проблем с производительностью и масштабируемостью при различном уровне нагрузки. В нашем случае в качестве тестирования производительности было протестировано заполнение одной из таблиц 100000 строк. После добавления индекса, можно сделать вывод, что тестирование прошло успешно.

6 Описание технологий и их применение в базе данных

6.1 Резервное копирования и восстановление

Резервная копия – это набор всех файлов, которые необходимы для восстановления корректной работы базы данных при частичном удалении файлов или нарушении структуры данных.

Для создания резервной копии базы данных был написан bat-файл, который создаёт файл с расширением bak по указанному пути.

Разработанный bat-файл представлен на листинге 6.1

SQLCMD -S (localdb)\MSSQLLocalDB -E -Q "BACKUP DATABASE Delivery TO DISK = 'C:\Users\Erik\Desktop\3course\Delivery\DBBackup\BackUpDelivery\_Sun.bak' WITH INIT, NOFORMAT, SKIP, NOUNLOAD"

Листинг 6.1 – Bat-файл для создания резервной копии базы данных

Для автоматизации резервного копирования базы данных использовался планировщик заданий Windows.

На рисунке 6.1 отображены задачи, которые создают резервные копии базы данных каждый день недели в 3:00.

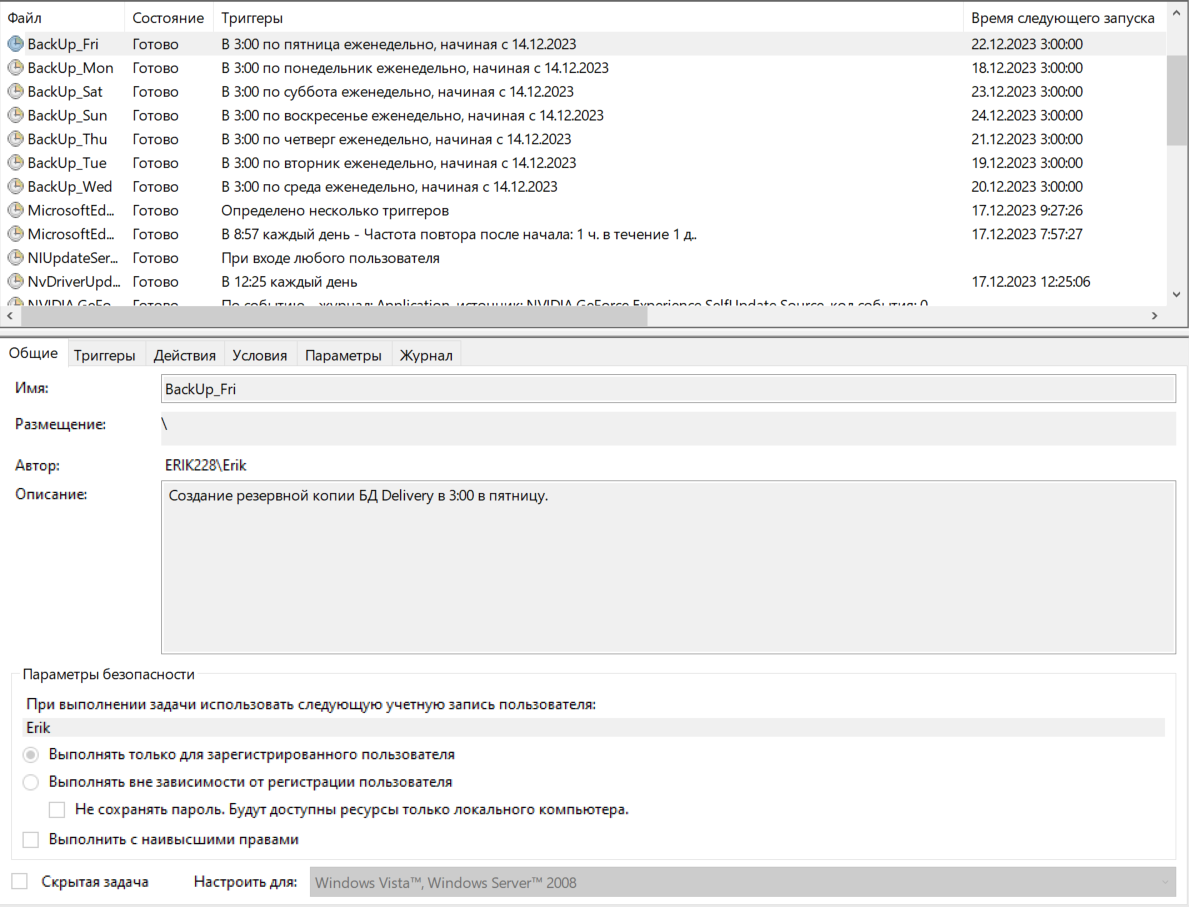


Рисунок 6.1 – Задачи планировщика Windows

Таким образом, при повреждении или удалении базы данных, мы сможем восстановить её при помощи bak файлов и встроенной функции Microsoft SQL Server Restore Database.

6.2 Настройка аудита

Аудит – это процесс мониторинга и регистрации всех действий, происходящих в базе данных. Целью аудита является обеспечение безопасности, целостности и конфиденциальности данных, а также контроль доступа к ним.

Во время аудита регистрируются различные события, такие как создание, изменение или удаление данных, выполнение запросов, авторизация пользователей и другие действия, связанные с базой данных. Эти записи обычно сохраняются в специальном журнале аудита (audit log) для последующего анализа.

Аудит позволяет отслеживать активности пользователей, выявлять нарушения безопасности, обнаруживать несанкционированный доступ или изменения данных, а также проводить анализ и расследование инцидентов. Также аудит может быть полезен для соблюдения требований законодательства и нормативных актов в отношении обработки и хранения данных. Настройка спецификации аудита базы данных представлена на рисунке 6.2.

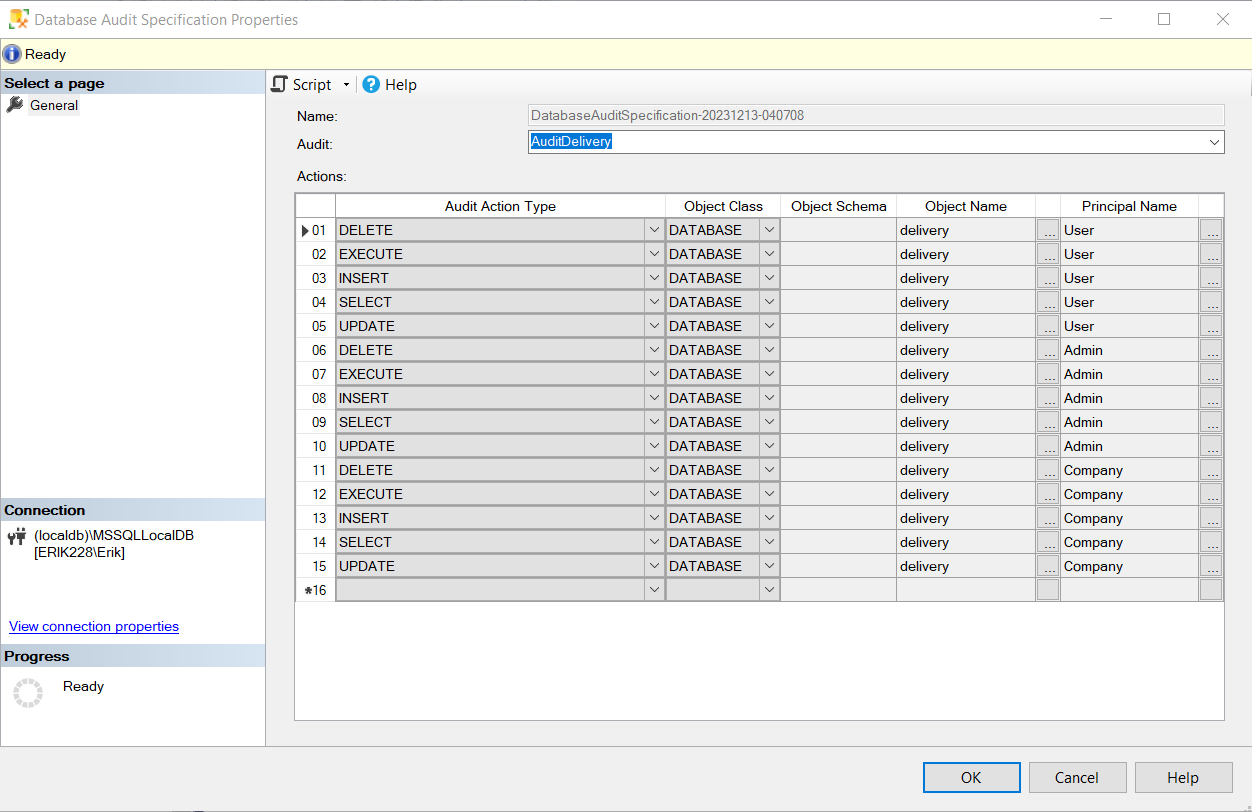


Рисунок 6.2 – Настройка спецификации аудита базы данных

Данная настройка позволяет отслеживать такие действия как: удаление, обновление, вставка, выполнение процедур пользователей User, Admin и Company базы данных.

Также была проведена настройка спецификации аудита сервера, что позволяет отслеживать все удачные и неудачные подключения к серверу.

6.3 Вывод по разделу

В данном разделе была описана настройка системы безопасности, а именно был настроен механизм автоматического создания резервных копий базы данных и аудит, который позволяет отслеживать все действия пользователей, что обеспечит безопасность базы данных.

7 Краткое описание приложения для демонстрации

Для демонстрации работы базы данных было разработано приложение на платформе ASP.NET Core, с использованием языка «C#».

Для работы с базой данных была использована технология ADO.NET. Все операции, такие как получение данных, их изменение и другие были реализованы с помощью процедур базы данных.

В приложении реализованы три роли: User, Admin и Company Admin, у каждой роли присутствует собственный функционал.

Веб-приложение реализует архитектурный паттерн MVC («Модель-Представление-Контроллер»), состоит из 16 представлений: «Корзина», «Профиль», «Регистрация», «Авторизация», «Заказы» и так далее, 9 контроллеров и 8 моделей.

8 Руководство пользователя

Приложение представляет собой web-сайт, с которым взаимодействуют пользователи. При запуске приложения открывается главная страница, на которой пользователь может увидеть список ресторанов, рисунок 8.1.

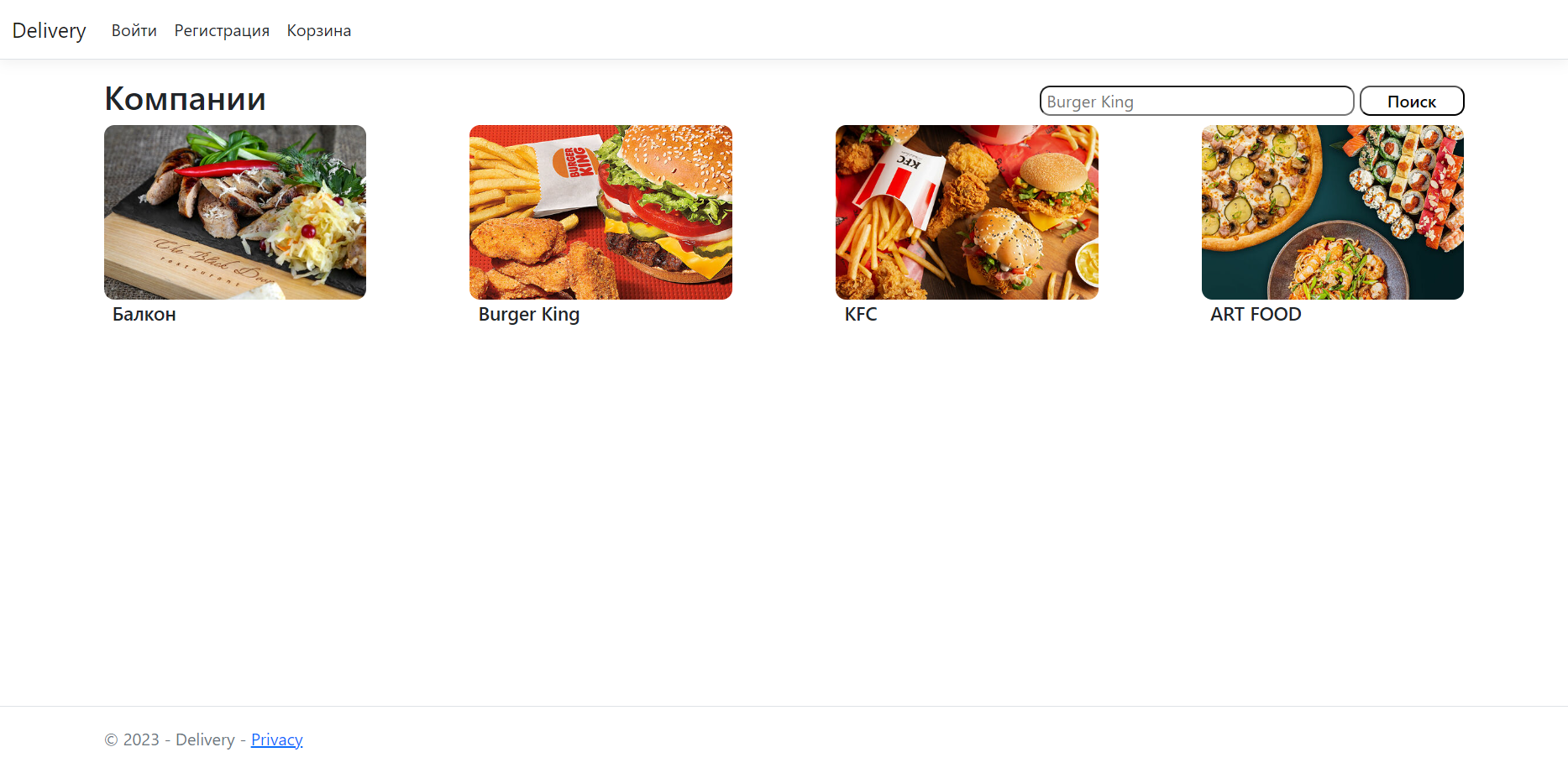


Рисунок 8.1 – Главная страница

Для того, чтобы зарегестрироваться на сайте необходимо нажать кнопку «Регистрация». После этого откроется страница регистрации, рисунок 8.2.

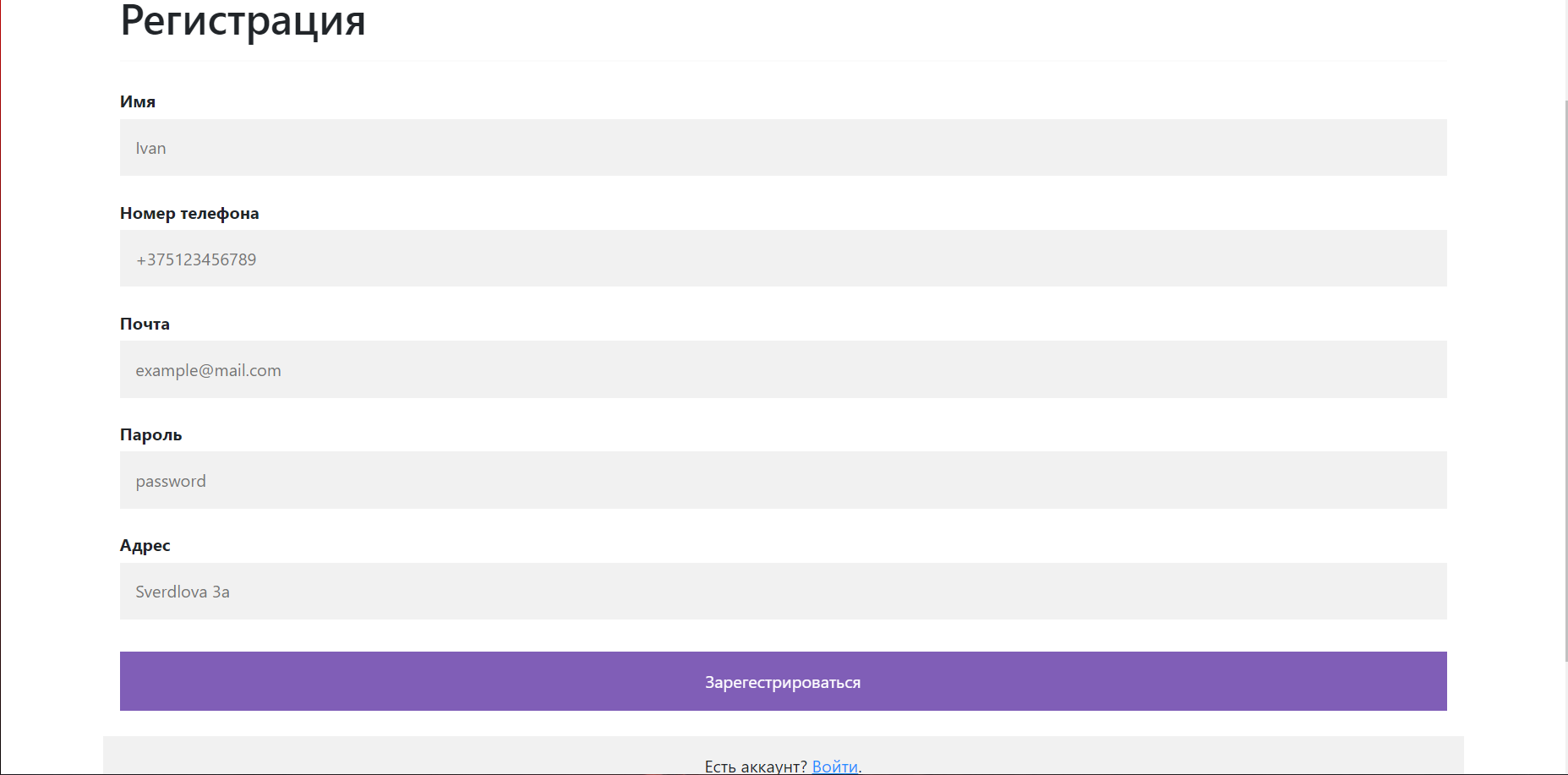


Рисунок 8.2 – Страница регистрации

В данной форме все поля являются обязательными. Поля с номером телефона, почтой и паролем проверяются на корректность и, при неверно введённых данных, отображают соответствующую ошибку на странице. Пример ошибки приведён на рисунке 8.3.

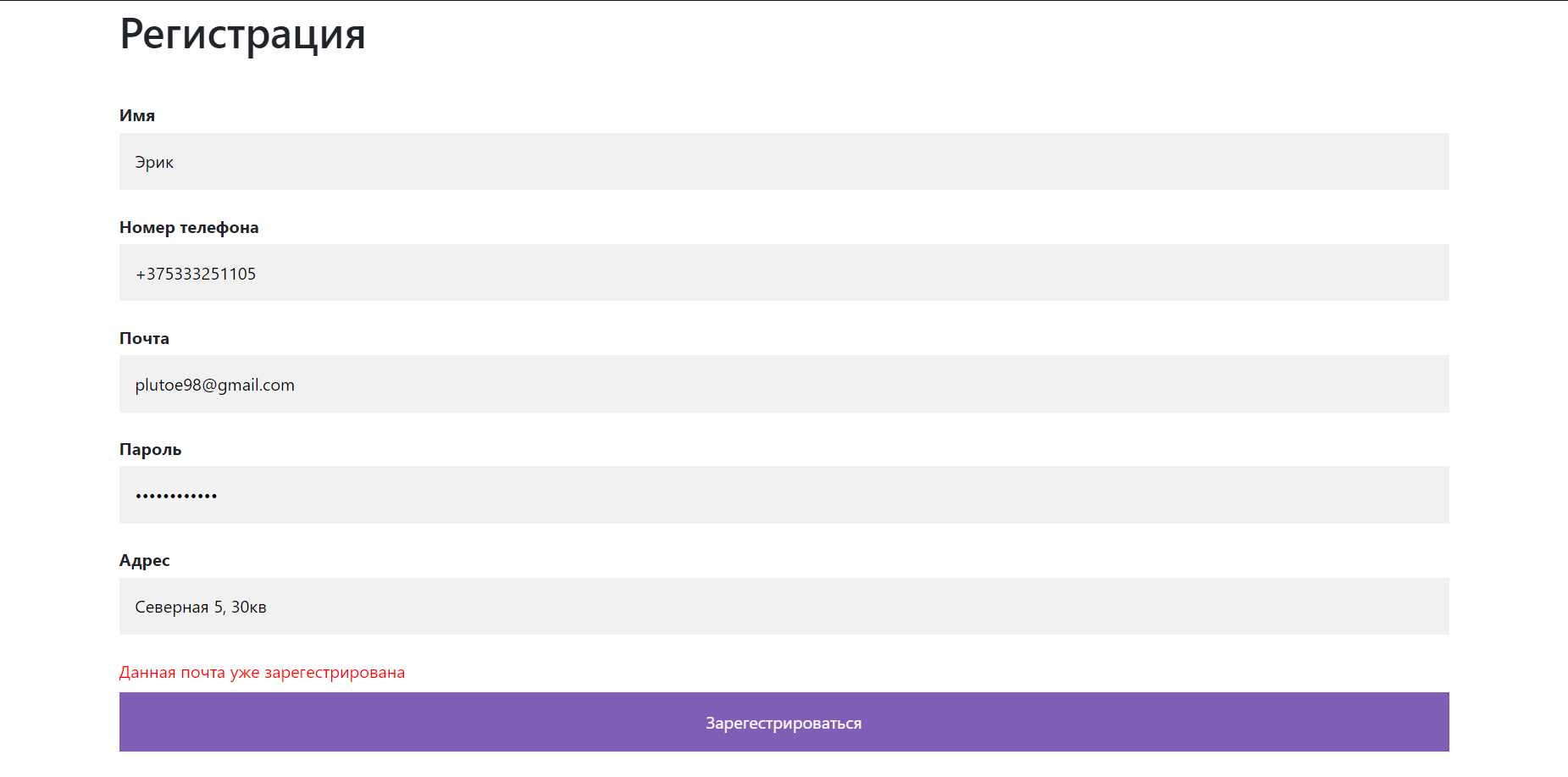


Рисунок 8.3 – Пример ошибки при регистрации

Если пользователь уже зарегестрирован, то он может нажать на кнопку «Войти» на любой странице веб-приложения, либо на кнопку «Есть аккаунт? Войти» на странице регистрации. После чего ему откроется страница авторизации, рисунок 8.4.

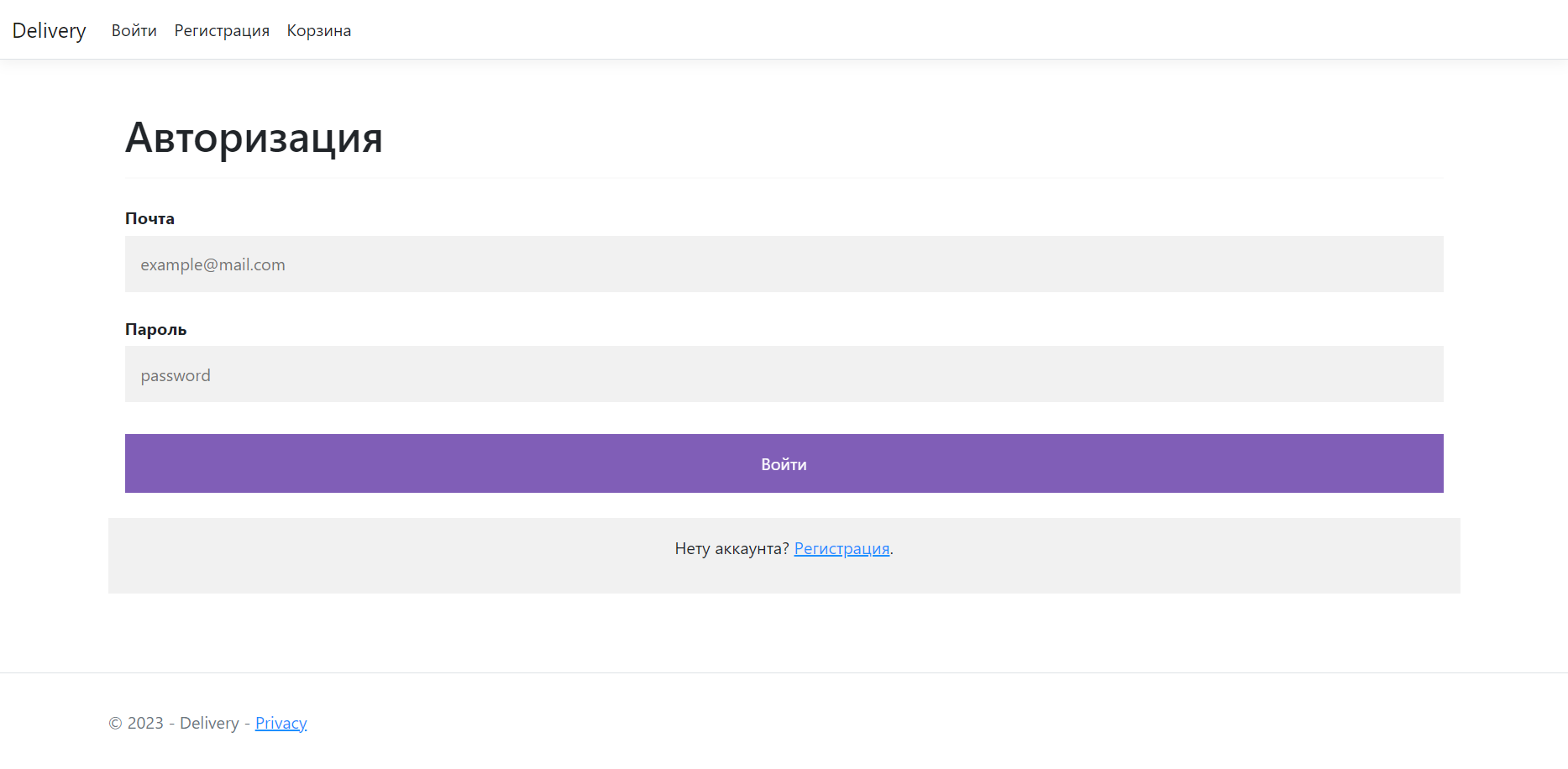


Рисунок 8.4 – Страница авторизации

На этой странице также все поля обязательны для ввода. После заполнения полей и нажатия кнопки «Войти», проверяется соответствие введённых данных с данными в базе данных, если они не соответствуют, будет выведена ошибка «Неверная почта или пароль» на странице, иначе – пользователь будет успешно авторизован и попадёт на главную страницу приложения.

В приложении присутствуют 3 роли: user, admin и company admin. Если пользователь является администратором, то, после успешной авторизации, он увидит соответствующее сообщение на страницах. Пример сообщения представлен на рисунке 8.5.

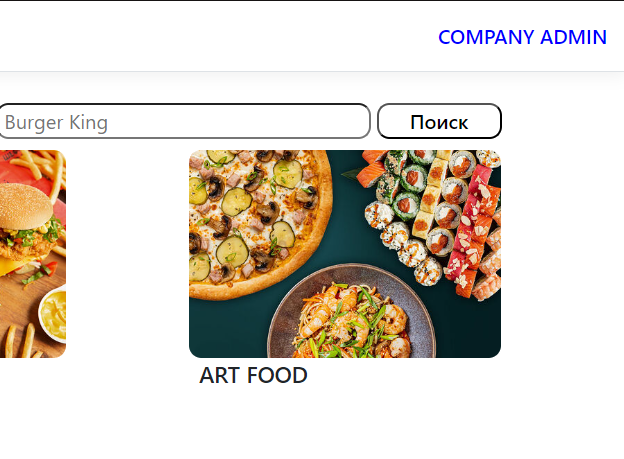


Рисунок 8.5 – Сообщение администратора компании на главной странице

Если пользователь не является администратором, то сообщения, указанного выше, не будет. После успешной авторизации, пользователь, не являющийся администратором, сможет открыть две новые страницы: «Мои заказы», в которой сможет просмотреть список активных оформленных заказов, и «Профиль», в которой пользователь может просмотреть и редактировать данные своего аккаунта, а также удалить его.

На главной странице пользователи могут выполнять поиск по названию ресторанов, для этого нужно ввести искомые символы в строку поиска и нажать кнопку «Поиск». Например, найдем рестораны в названии которых есть буква «K», рисунок 8.6.

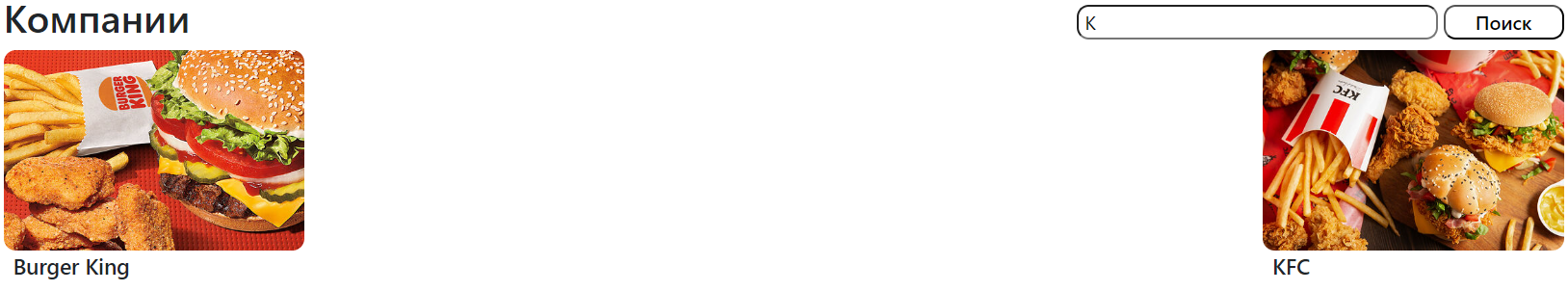


Рисунок 8.6 – Поиск компаний

При нажатии на обложку одного из ресторанов откроется страница ресторана, представляющая собой каталог товаров этого ресторана, рисунок 8.7.

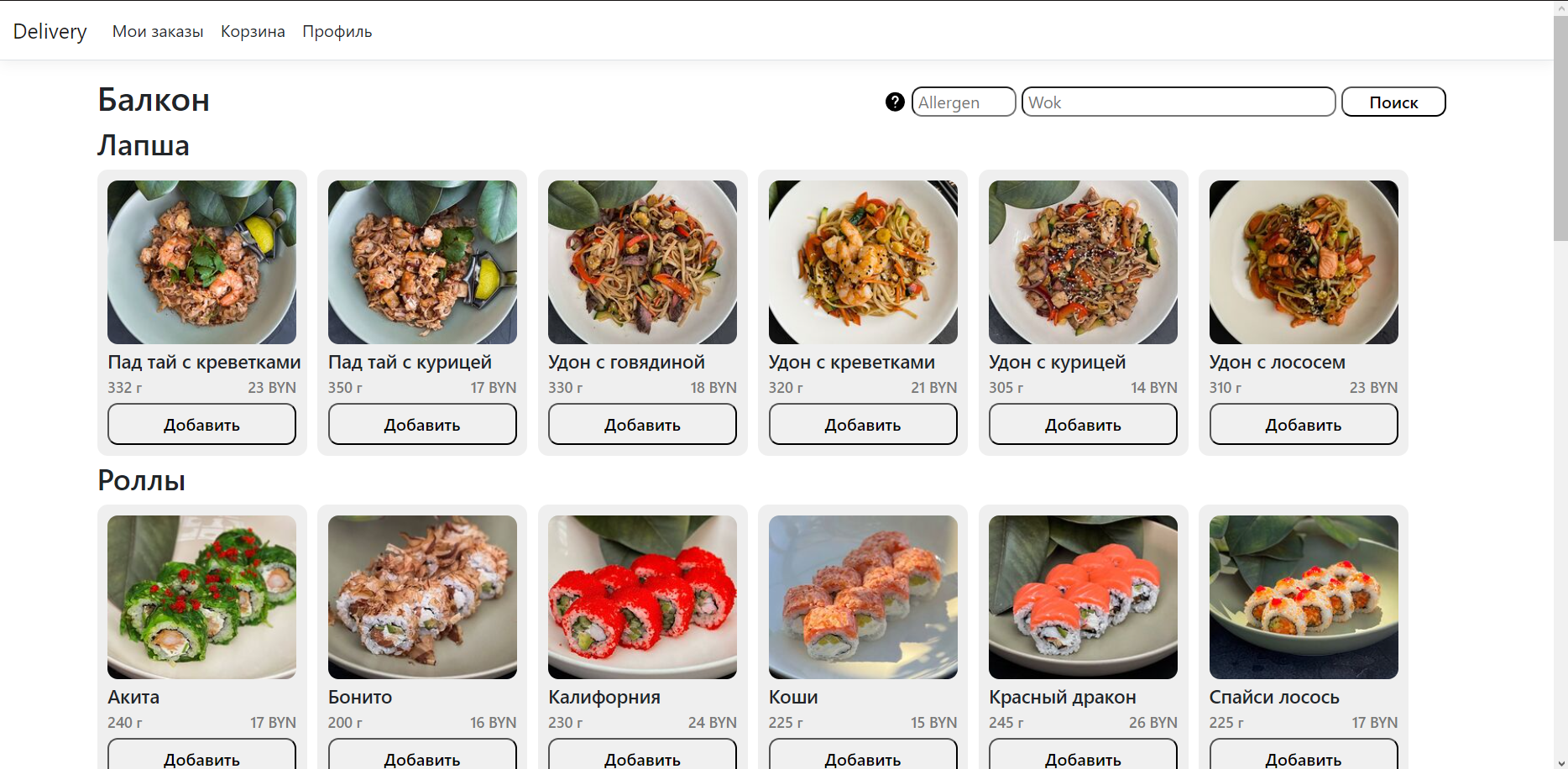


Рисунок 8.7 – Страница ресторана

Все товары разделены на категории, при этом, если в категории отсутствуют товары, обычный пользователь её не увидит. При нажатии на обложку товара, откроется окно с дополнительной информацией о нём, рисунок 8.8.

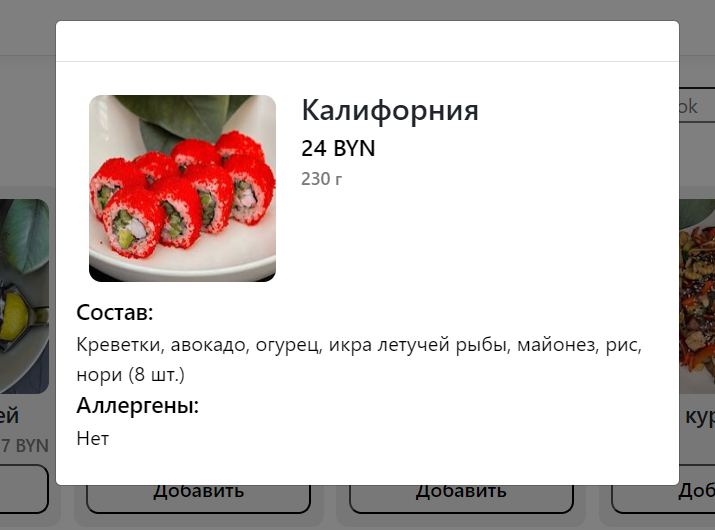


Рисунок 8.8 – Окно с дополнительной информацией о товаре

Также на этой странице пользователь может выполнять поиск по аллергенам и названиям товаров, для этого он должен ввести номера аллергенов и искомые символы в соответствующие поля, чтобы увидеть номера аллергенов и их названия пользователь должен навести курсор мыши на подсказку в виде вопросительного знака возле строки поиска. Пример, в котором выполняется поиск товаров без аллергена «Рыба», идентификатор 4, представлен на рисунке 8.9.

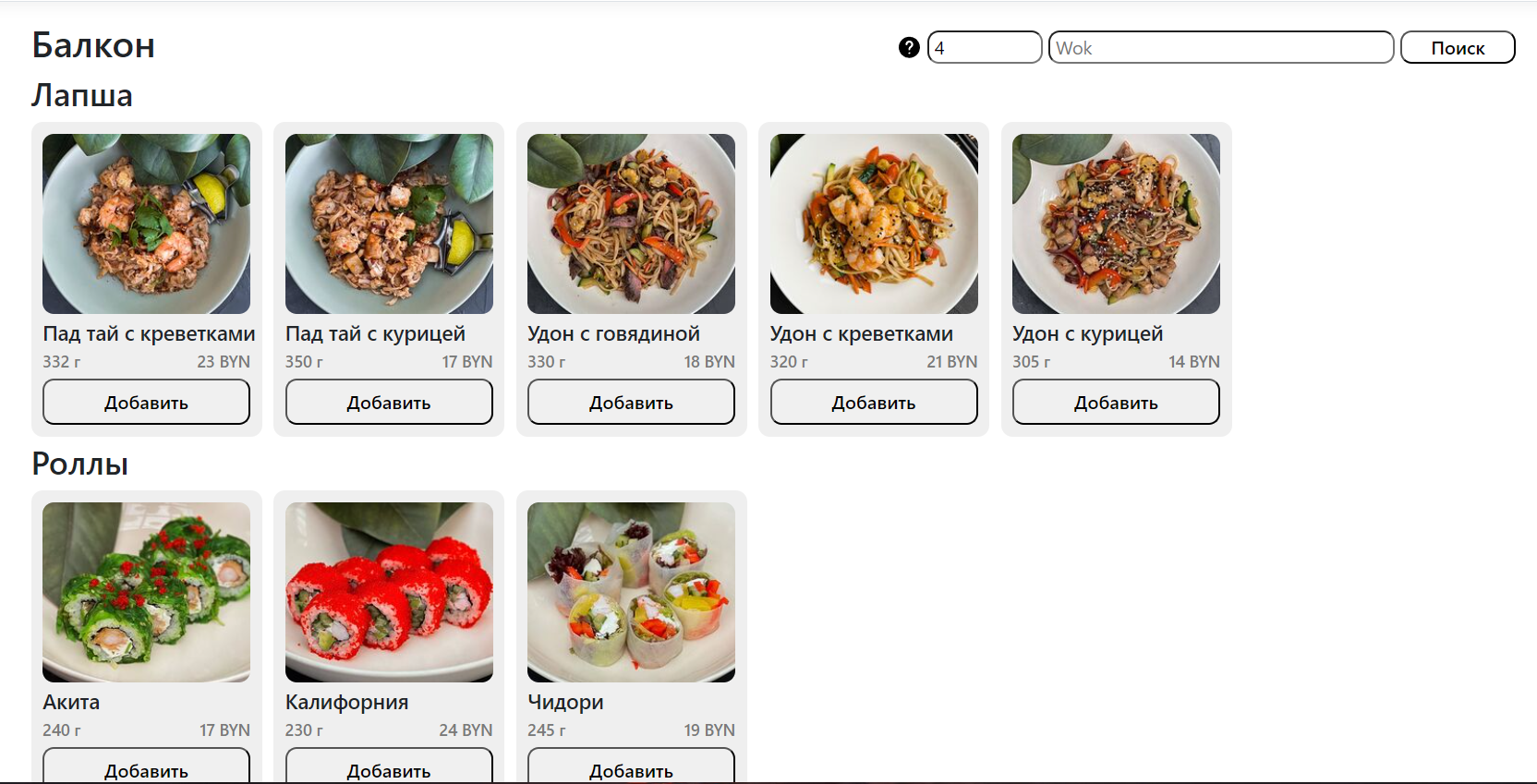


Рисунок 8.9 – Поиск товаров

Чтобы добавит товар в корзину пользователь должен нажать на кнопку «Добавить» соответствующего товара. После добавления товара в корзину кнопка этого товара поменяет свой текст на «Убрать» и, при нажатии на эту кнопку, товар уберётся из корзины.

Чтобы оформить заказ пользователь должен зайти на страницу корзины, для этого он должен нажать на кнопку «Корзина» на любой из страниц приложения. На этой странице пользователь увидит список ранее добавленных товаров, краткую информацию о них, их количество, общую стоимость заказа, способ оплаты и кнопку «Оформить заказ». При нажатии на кнопки «+» и «-» напротив товара будет соответственно увеличиваться или уменьшаться количество товара, если количество станет равно нулю, товар удалиться из корзины. Чтобы изменить способ оплаты нужно нажать на соответствующий элемент формы, после чего пользователь сможет выбрать один из двух способов («Наличные» или «Карта»). Чтобы завершить оформление заказа пользователь должен нажать на кнопку «Оформить заказ», после чего он попадёт на страницу «Мои заказы», где появится только что оформленный заказ. Страница корзины с двумя добавленными товарами представлена на рисунке 8.10.

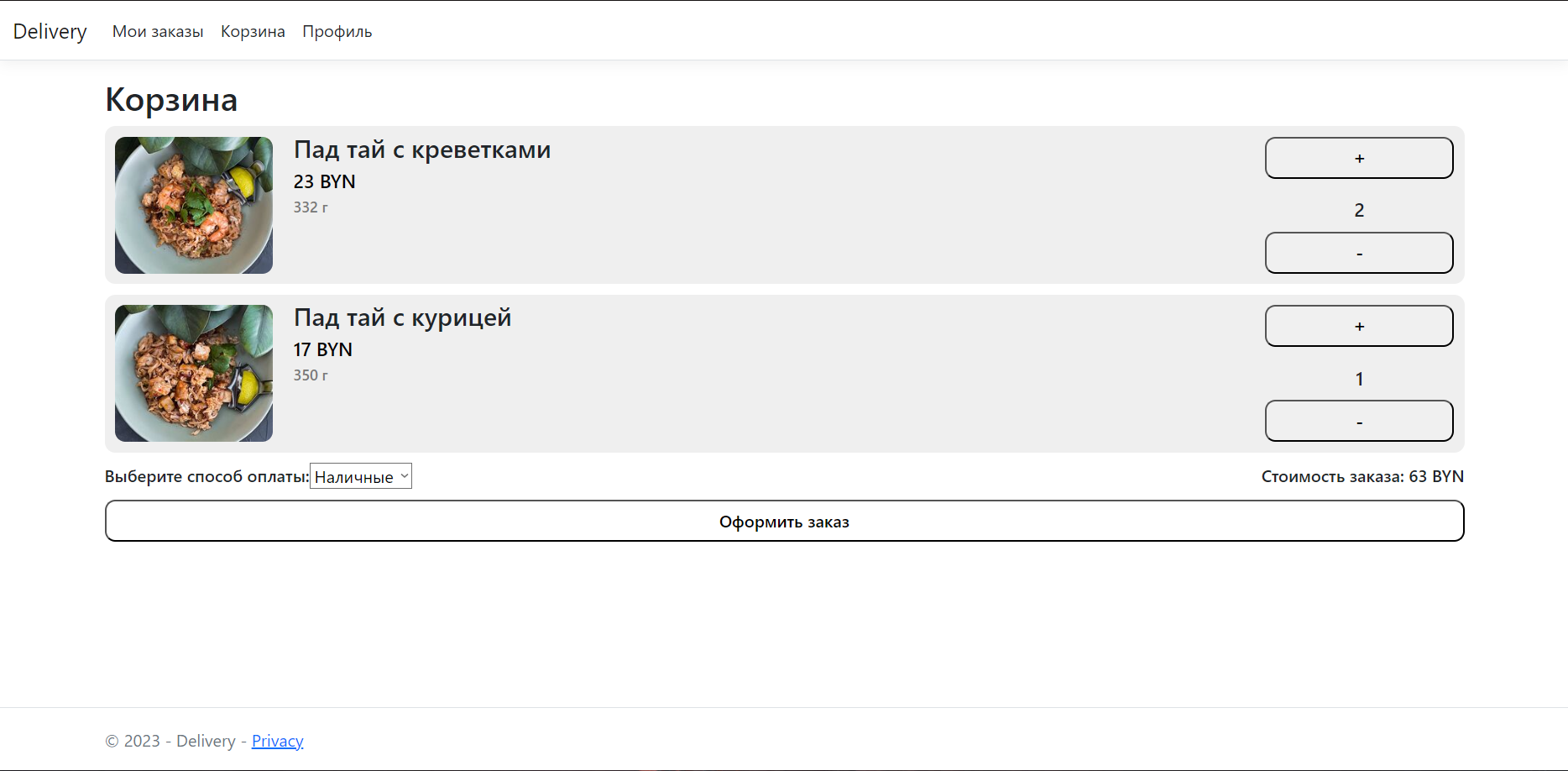


Рисунок 8.10 – Страница корзины

На страницу «Мои заказы» пользователь может попасть, нажав на соответствующую кнопку или после оформления заказа, рисунок 8.11.

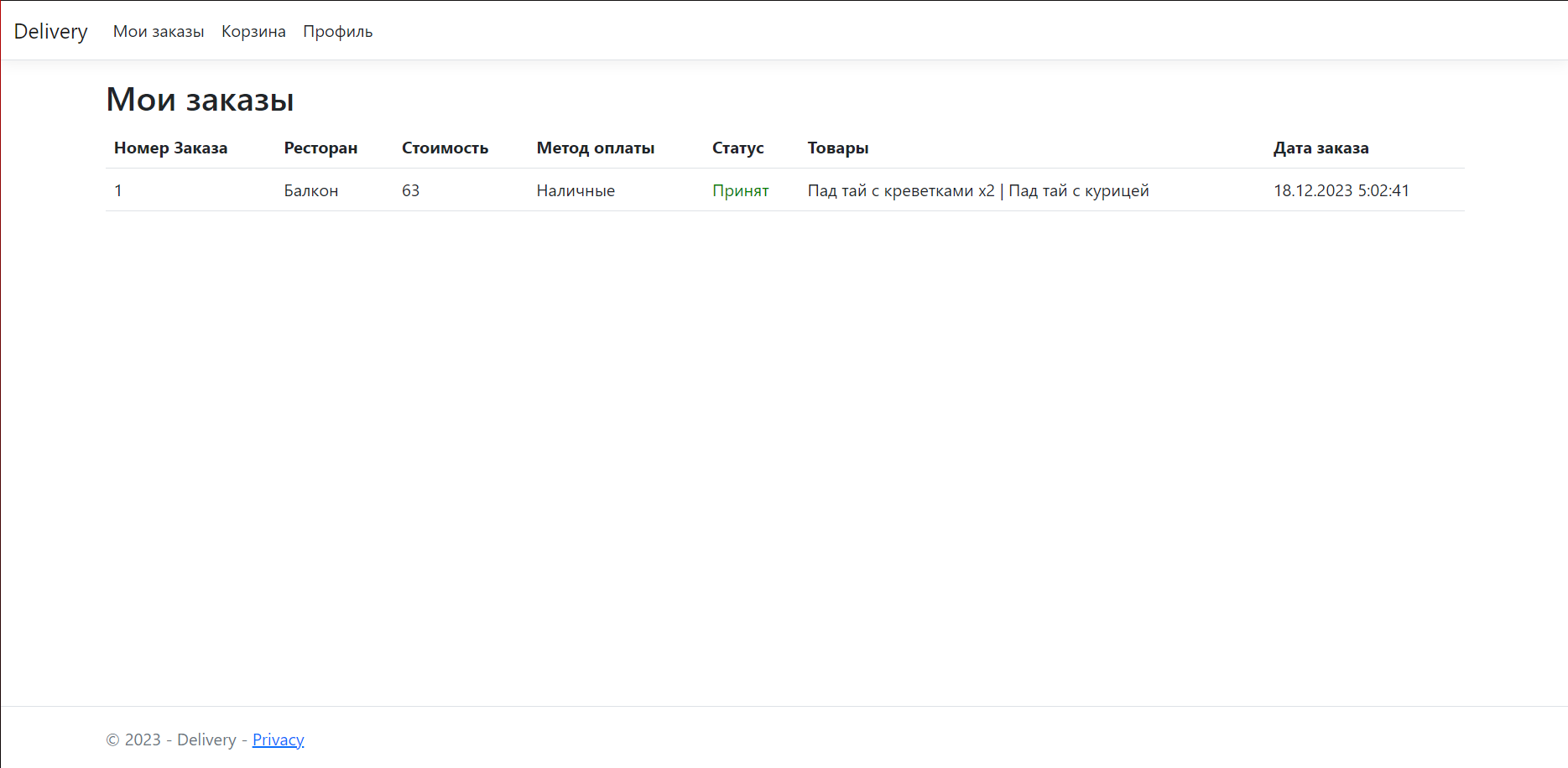


Рисунок 8.11 – Страница с заказами пользователя

На данной странице пользователь может увидеть список всех его заказов и информацию о них.

Если пользователь является администратором, то после успешной авторизации сможет открыть четыре новые страницы: «Заказы», в которой сможет просмотреть список активных оформленных заказов всех ресторанов, «История», в которой сможет просмотреть дату и информацию о создании и каждом изменении заказа, «Аккаунты», в которой администратор сможет увидеть список всех аккаунтов приложения, их данные, а также изменить их привилегии или удалить, «Профиль», в которой администратор может просматривать и редактировать данные своего аккаунта. Чтобы попасть на эти страницы необходимо нажать на соответствующие кнопки на любой из страниц приложения. Страница «История» представлена на рисунке 8.12.

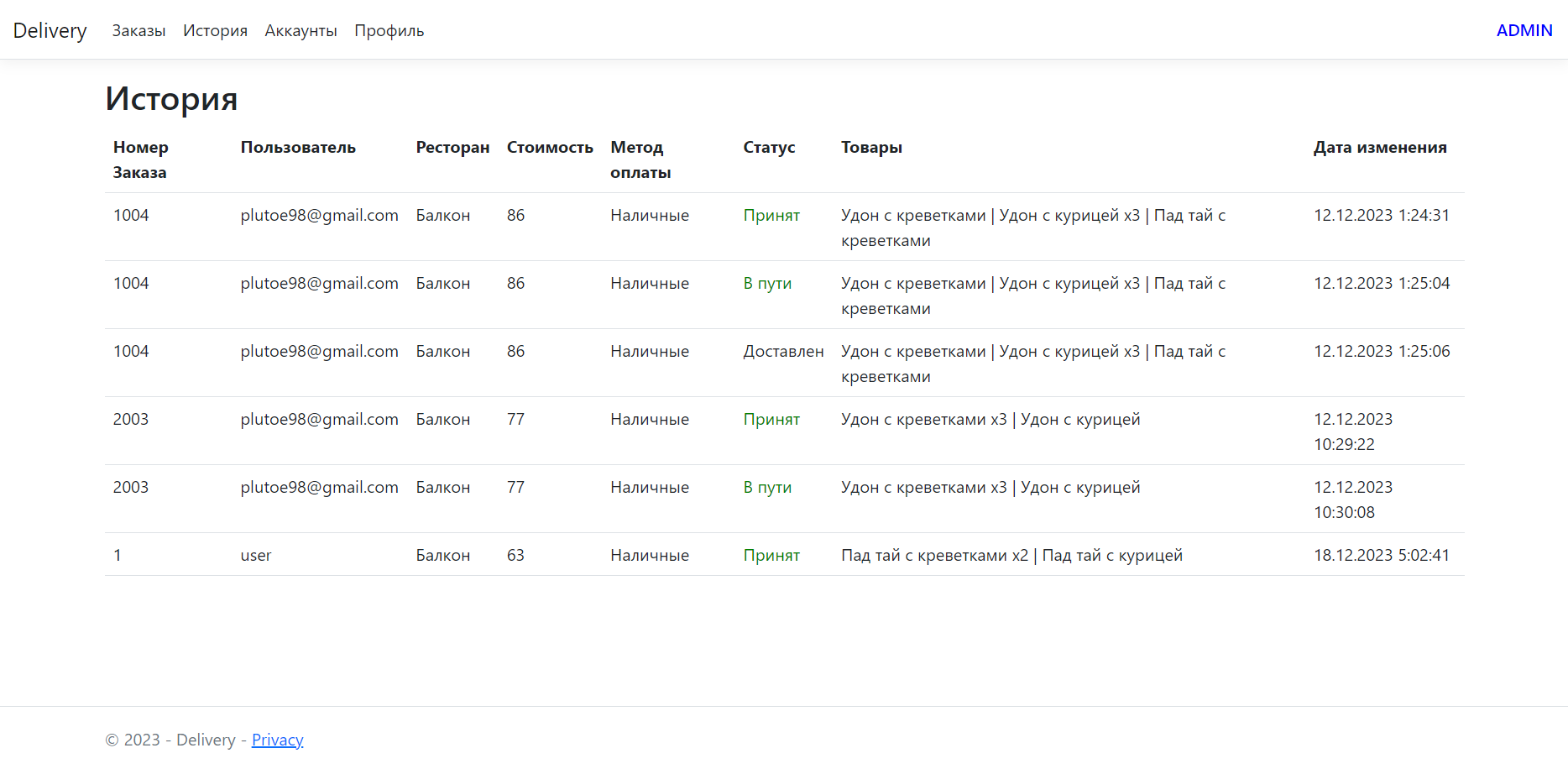


Рисунок 8.12 – Страница с историей заказов

Эта страница предназначена для отслеживания создания и изменения статуса заказов.

Страница «Аккаунты» представлена на рисунке 8.13.

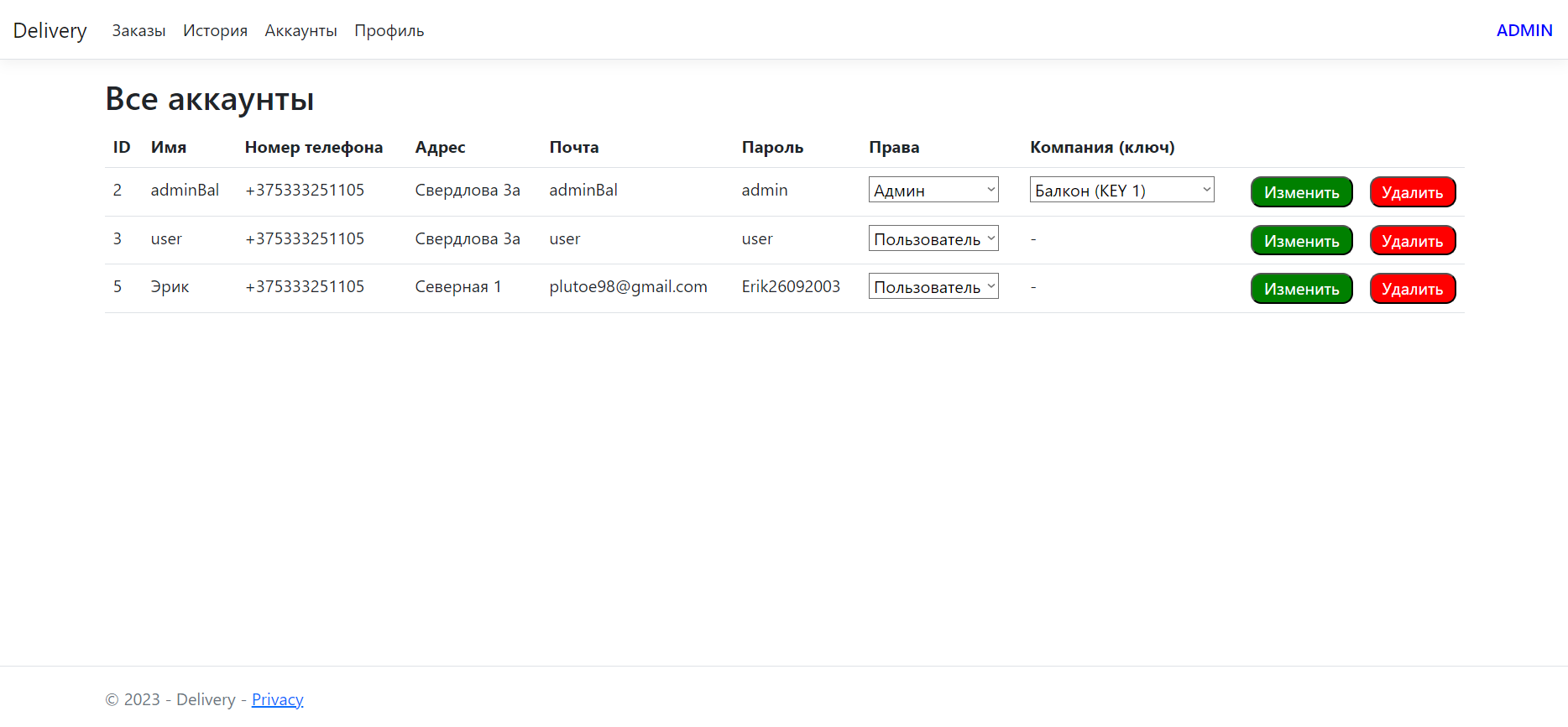


Рисунок 8.13 – Страница с аккаунтами приложения

На этой странице администратор может просматривать и управлять аккаунтами приложения. Чтобы изменить права пользователя или его уровень привилегий необходимо нажать на соответствующие кнопки формы. Чтобы сохранить изменения администратор должен нажать на кнопку «Изменить». Если необходимо удалить аккаунт нужно нажать на кнопку «Удалить».

Если пользователь является администратором компании, то после успешной авторизации сможет открыть четыре новые страницы: «Заказы», в которой сможет просмотреть список активных оформленных заказов своего ресторана, «История», в которой сможет просмотреть дату и информацию о создании и каждом изменении заказов своей компании, «Профиль», в которой администратор может просматривать и редактировать данные своего аккаунта. Чтобы попасть на эти страницы необходимо нажать на соответствующие кнопки на любой из страниц приложения. Страница «Заказы» представлена на рисунке 8.14.

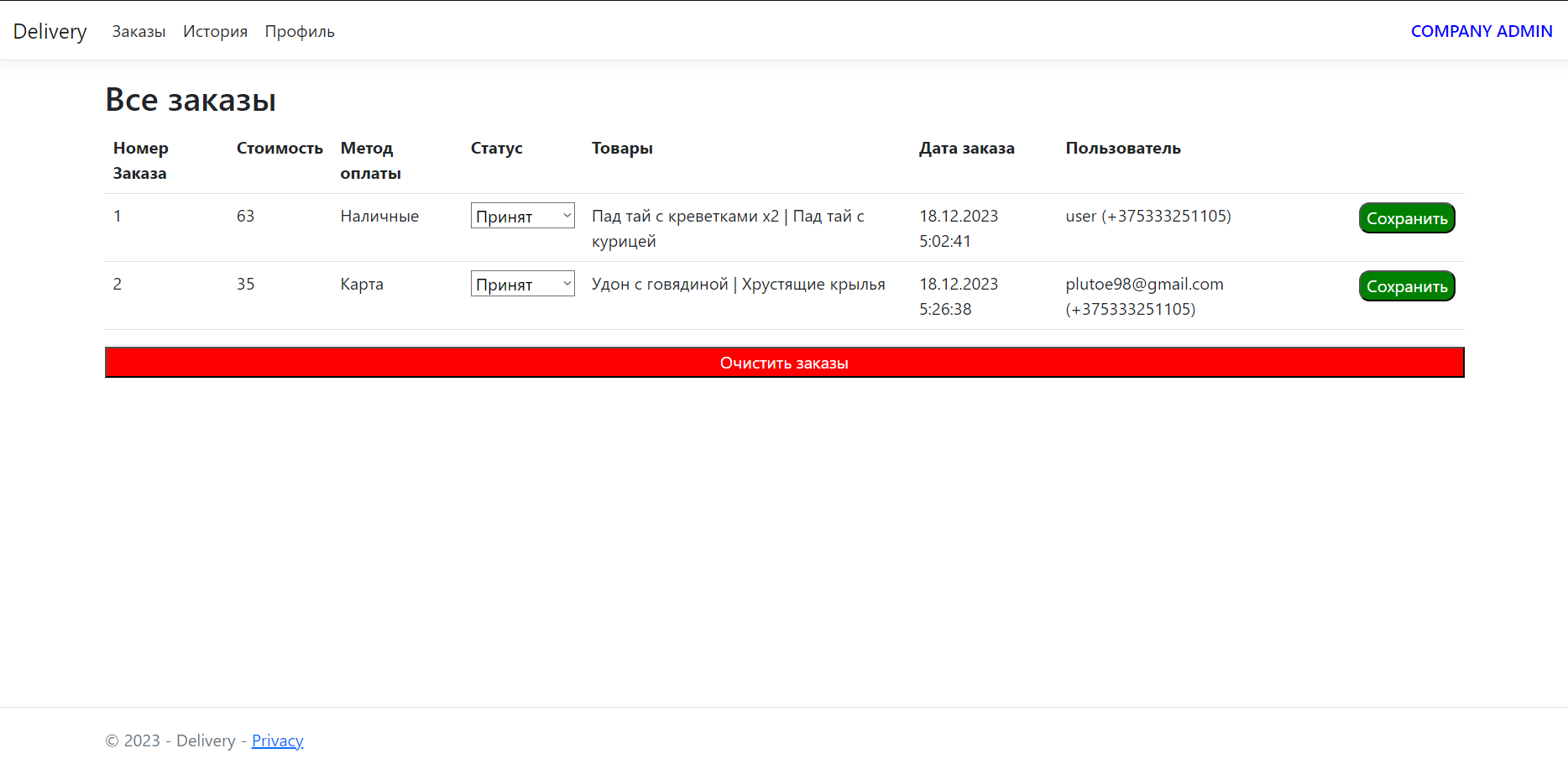


Рисунок 8.14 – Страница заказами компании

На данной странице администратор может изменить статус заказа, для этого необходимо выбрать новый статус в соответствующем элементе формы и нажать на кнопку «Сохранить». Также на этой странице администратор может очистить заказы компании, для этого ему необходимо нажать на кнопку «Очистить заказы».

Заключение

В данном курсовом проекте была разработана база данных для сервиса по доставке еды с настройкой системы безопасности базы данных.

Были проанализированы достоинства и недостатки аналогов, что позволило определить необходимый функционал, набор сущностей и их связи.

Был реализован следующий функционал: роли администратора, администратора компании и пользователя, для каждого из которых был реализован собственный функционал, управление пользователями (регистрация, авторизация, редактирование, удаление), управление компаниями, категориями и товарами, включающее в себя их добавление, редактирование и удаление, управление заказами, их оформление, изменение статуса и удаление, поиск товаров по названию и аллергену, а также поиск компаний по названию. Были реализованы импорт и экспорт данных таблиц в JSON формат и заполнение таблицы на 100 000 записей (таблица товаров).

Для демонстрации работы базы данных было разработано приложение на платформе ASP.NET Core, с использованием языка «C#», а также необходимые функции, триггеры и процедуры для исправного взаимодействия приложения с базой данных.

Были обработаны ошибки, которые могут возникнуть на стороне приложения или базы данных.

Был настроен аудит и автоматическое резервное копирование базы данных, что обеспечит её безопасность.

Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

Список литературы

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.10.2021.

2. Microsoft.com [Электронный ресурс] / Режим доступа: https://learn.microsoft.com – Дата доступа: 27.10.2023.

3. Настройка регулярного резервного копирования БД MS SQL Server [Электронный ресурс] / Режим доступа: https://www.alta.ru/support/51/ – Дата доступа: 16.10.2023.

4. Аудит в SQL Server [Электронный ресурс] / Режим доступа: http://www.t-sql.ru/post/audit.aspx – Дата доступа: 20.10.2021.

5. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2023

ПРИЛОЖЕНИЕ A

|  |
| --- |
| CREATE TABLE [dbo].[User](  [IDUser] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](50) NOT NULL,  [PhoneNo] [nvarchar](50) NOT NULL,  [Address] [nvarchar](50) NOT NULL,  [Email] [nvarchar](50) NOT NULL,  [Password] [varbinary](500) NOT NULL,  [IsAdmin] [bit] NOT NULL,  [AdminKey] [int]  )  CREATE TABLE [dbo].[Order](  [IDOrder] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,  [IDUser] [int] NOT NULL FOREIGN KEY REFERENCES [User](IDUser),  [IDCompany] [int] NOT NULL FOREIGN KEY REFERENCES [Company](IDCompany),  [OrderDate] [datetime] NOT NULL,  [Status] [nvarchar](50) NOT NULL,  [PaymentMethod] [nvarchar](50) NOT NULL,  [TotalPrice] [numeric](8, 2) NOT NULL,  [Products] [nvarchar](300) NULL  )  CREATE TABLE [dbo].[History](  [IDOrder] [int] NOT NULL,  [User] [nvarchar](50) NOT NULL,  [Company] [nvarchar](50) NOT NULL,  [OrderDate] [datetime] NOT NULL,  [Status] [nvarchar](50) NOT NULL,  [PaymentMethod] [nvarchar](50) NOT NULL,  [TotalPrice] [numeric](8, 2) NOT NULL,  [Products] [nvarchar](300) NULL,  )   CREATE TABLE [dbo].[Category](  [IDCategory] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](50) NOT NULL,  [IDCompany] [int] NOT NULL FOREIGN KEY REFERENCES Company(IDCompany)  )  CREATE TABLE [dbo].[Allergen](  [IDAllergen] [int] PRIMARY KEY NOT NULL,  [Name] [nvarchar](50) NOT NULL,  )  CREATE TABLE [dbo].[Company](  [IDCompany] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](50) NOT NULL,  [ImagePath] [nvarchar](150) NOT NULL  )  CREATE TABLE [dbo].[Product](  [IDProduct] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](30) NOT NULL,  [ImagePath] [nvarchar](100) NOT NULL,  [Description] [ntext] NOT NULL,  [Price] [numeric](8, 2) NOT NULL,  [IDCategory] [int] NOT NULL FOREIGN KEY REFERENCES Category(IDCategory),  [Gram] [int] NOT NULL,  [Allergens] [nvarchar](50) NULL,  ) |

Листинг 1 – Скрипты создания таблиц

ПРИЛОЖЕНИЕ Б

|  |
| --- |
| CREATE procedure [dbo].[DeleteProduct]  @id int  as  begin try  delete Product where IDProduct = @id  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  CREATE procedure [dbo].[DeleteUser]  @id int  as  begin try  if not exists (select \* from [User] where IDUser = @id)  RAISERROR ('No such user', 0, 1)  delete [Order] where IDUser = @id  delete [User] where IDUser = @id  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  CREATE procedure [dbo].[UpdateRights]  @id int, @isadmin bit, @key int  as  begin try  if not exists (select \* from [User] where IDUser = @id)  RAISERROR ('No such user', 0, 1)  update [User] set IsAdmin = @isadmin, AdminKey = @key where IDUser = @id  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  CREATE procedure [dbo].[ChangeStatus]  @id int, @status nvarchar(50)  as  BEGIN TRY  if @status != 'Принят' and @status != 'В пути' and @status != 'Доставлен' and @status != 'Отменён'  RAISERROR ('Incorrect status', 16, 1)  if not exists (select \* from [Order] where IDOrder = @id)  RAISERROR ('No such order', 0, 1)  update [Order] set Status = @status where IDOrder = @id  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  CREATE procedure [dbo].[CreateCategory]  @name nvarchar(50), @company int  as  begin try  insert Category(Name, IDCompany) values (@name, @company)  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  CREATE procedure [dbo].[CreateCompany]  @name nvarchar(50), @path nvarchar(150)  as  begin try  insert into [Company](Name, ImagePath) values  (@name, @path);  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH |

Листинг 2 – Скрипты создания процедур

ПРИЛОЖЕНИЕ В

|  |
| --- |
| CREATE function [dbo].[CheckEmailFunc] (@email nvarchar(50))  returns bit  as  begin  declare @valid bit  if (select COUNT(\*) from [User] where Email = @email) = 0  set @valid = 1  else  set @valid = 0  return @valid  end  CREATE function [dbo].[CompanyAveragePrice](@CompanyId int)  returns numeric(8, 2)  as  begin  return (  select AVG(Price) from [Product]  inner join Category  on Category.IDCategory = Product.IDCategory  where IDCompany = @CompanyId)  end  CREATE function [dbo].[AveragePrices]()  returns table  as  return select [Name] as [Ресторан], [dbo].CompanyAveragePrice(IDCompany) as [Средняя цена продукта] from [Company] |

Листинг 3 – Скрипты создания функций

ПРИЛОЖЕНИЕ Г

|  |
| --- |
| CREATE TRIGGER [dbo].[History\_Trigger]  on [dbo].[Order]  AFTER INSERT, UPDATE  as  insert into History  select IDOrder, [User].Email, [Company].Name, GETDATE(), Status, PaymentMethod, TotalPrice, Products from inserted  inner join [User]  on inserted.IDUser = [User].IDUser  inner join [Company]  on inserted.IDCompany = [Company].IDCompany |

Листинг 4 – Скрипт создания триггера