**Введение**

Объект

«Рамен и точка – магазин продажи рамена»

Актуальность

Растущая популярность рамена: Рамен переживает настоящий гастрономический бум, и спрос на аутентичный и качественный рамен продолжает расти. Удобство онлайн-заказа: В условиях современного темпа жизни онлайн-магазин позволяет любителям рамена наслаждаться любимым блюдом, не выходя из дома или офиса. Охват широкой аудитории: Онлайн-формат позволяет охватить более широкую географию, чем стационарный ресторан, доставляя рамен в районы, где он может быть менее доступен. Персонализация и разнообразие: Онлайн-платформа позволяет предлагать широкий ассортимент раменов с возможностью выбора ингредиентов и степени остроты, удовлетворяя индивидуальные предпочтения. Потребители все больше ценят аутентичные продукты и кулинарные впечатления, а "Рамен и точка" фокусируется на качестве и традиционных рецептах. Доступность ингредиентов: Современные логистические возможности позволяют обеспечить доступ к свежим и качественным ингредиентам, необходимым для приготовления настоящего рамена.

**ГЛАВА 1. ПРЕДМЕТНАЯ ОБЛАСТЬ**

* 1. **Исходная постановка задачи**

Цель

Разработать приложение «Рамен и точка»

Задачи

1. Описать функциональные возможности приложения «Рамен и точка»

2. Проанализировать удобство использования приложения для различных категории пользователей.

3. Оценить влияние приложения формирования предпочтений и развития работоспособности онлайн ресторана.

4. Разработать рекомендации по улучшению приложения и его дальнейшему развитию.

Исходные данные

1. Продукты
2. Категории
3. Акции и скидки
4. Корзина
5. Пользователи

Выходные данные

1. Отзывы пользователей
2. Заказы

## **Среда разработки**

ОС – Windows 7/10/11

Среда разработки и язык программирования – Visual Studio, C#, СУБД: SQL Server.

Требования к среде выполнения

1. Поддержка современных версий операционных систем;
2. Обеспечение плавной работы приложения;
3. Поддержка хранения структурированных данных;
4. Возможность интеграции с внешними сервисами;
   1. **Техническое задание**

УТВЕРЖДЕНО

-022А 01.М-ЛУ к Договору № 2 от 02.04.06

Разработка информационной системы «Фитнес Клуб»

Техническое задание

02068120.508100.027-02 2А 01.М

Листов 5

2024

**ОГЛАВЛЕНИЕ**

1.ВВЕДЕНИЕ 6

2.ВХОДНЫЕ И ИСХОДНЫЕ ДАННЫЕ 7

3.АЛГОРИТМ РАБОТЫ ПРОГРАММЫ 8

**ВВЕДЕНИЕ**

Информационная система «Рамен и точка» – объект разработки, предназначенный для автоматизации производственных и управленческих процессов ресторана по продаже рамена. Система обеспечивает учет продуктов и клиентов, формирование заказов и отзывов. Приложение применимо на персональных компьютерах с ОС Windows 10/11

**2.ВХОДНЫЕ И ИСХОДНЫЕ ДАННЫЕ**

Входные данные содержат в себе:

- Данные о клиентах.

- Данные о продуктах.

- Данные о категориях.

- Данные о платежах..

- Данные о заказах.

Исходные данные:

- Отзывы клиентов.

- Данные о приобретенных продуктов

- Данные о корзине покупателя.

**3.АЛГОРИТМ РАБОТЫ ПРОГРАММЫ**

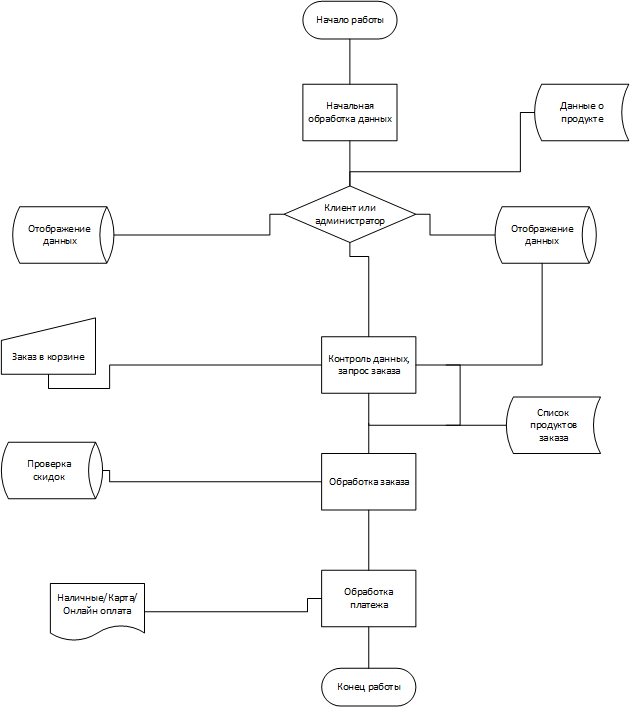


Рис. 1 Блок-схема (алгоритм программы)

**1.4 Руководство по стилю**

Общие требования:

Не допускать орфографических и грамматических ошибок.

Размер окон (форм) и кнопок

Титульная форма, форма авторизации, администратора и пользователя размер – 515x350, формы редактирования, просмотра и табличные формы – 850x450, Основной размер кнопок: 75x23 (По умолчанию), размер может зависеть от содержания. (Content)

Использование логотипа

Логотип используется в каждой форме слева в верхнем углу экрана, также используется как икона.

Шрифт

Шрифт Times New Roman. Размер шрифтов

14 Пт.

Персонализация элементов приложения

|  |  |
| --- | --- |
| #F8F0E3 | Основной цвет |
| #E27D60 | Акцентный цвет |
| #6E8898 | Дополнительный цвет |
| #333333 | Текст |

**1.5 ER – Диаграмма**

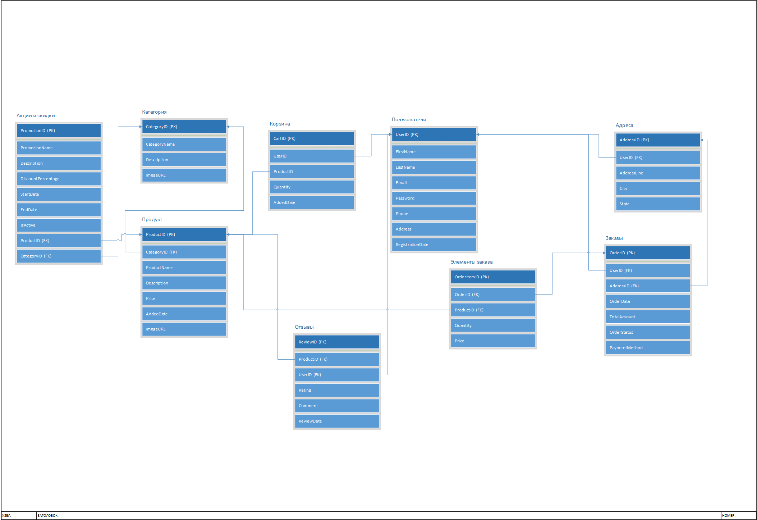


Рис. 2 ER – Диаграмма

**1.6 UML – Диаграммы**

**1.6.1 UML – Диаграмма прецендентов**

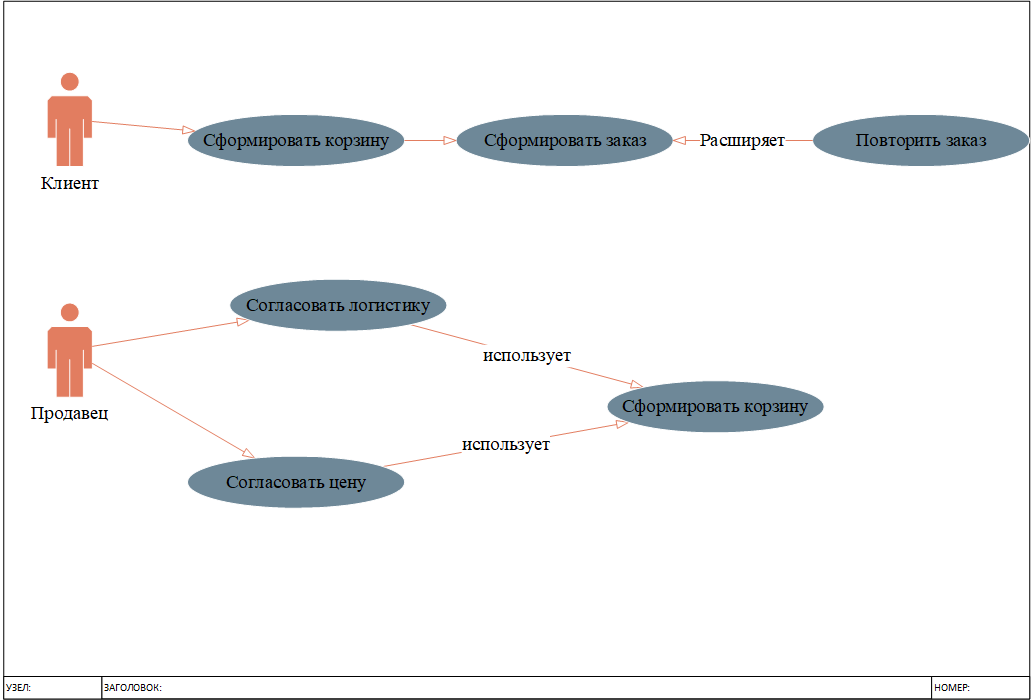


Рис. 3 ER – Диаграмма прецендентов

**1.6.2 UML – Диаграмма деятельности**

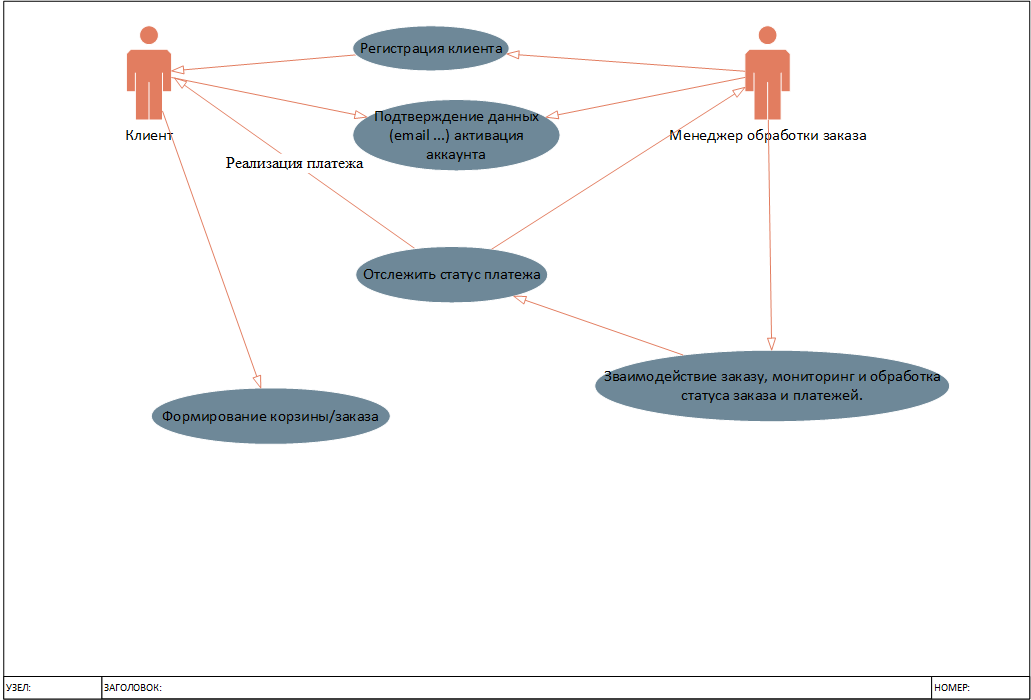


Рис. 4 UML – Диаграмма деятельности

**1.6.3 UML – Диаграмма взаимодействия**

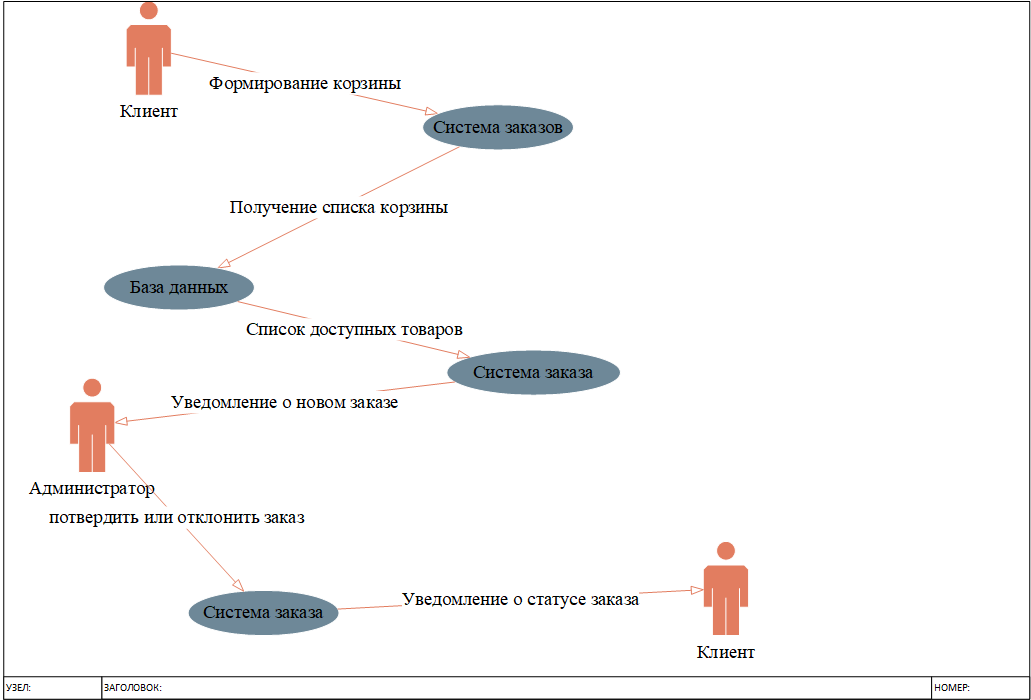


Рис. 5 UML – Диаграмма взаимодействия

**1.6.4 UML – Диаграмма классов**

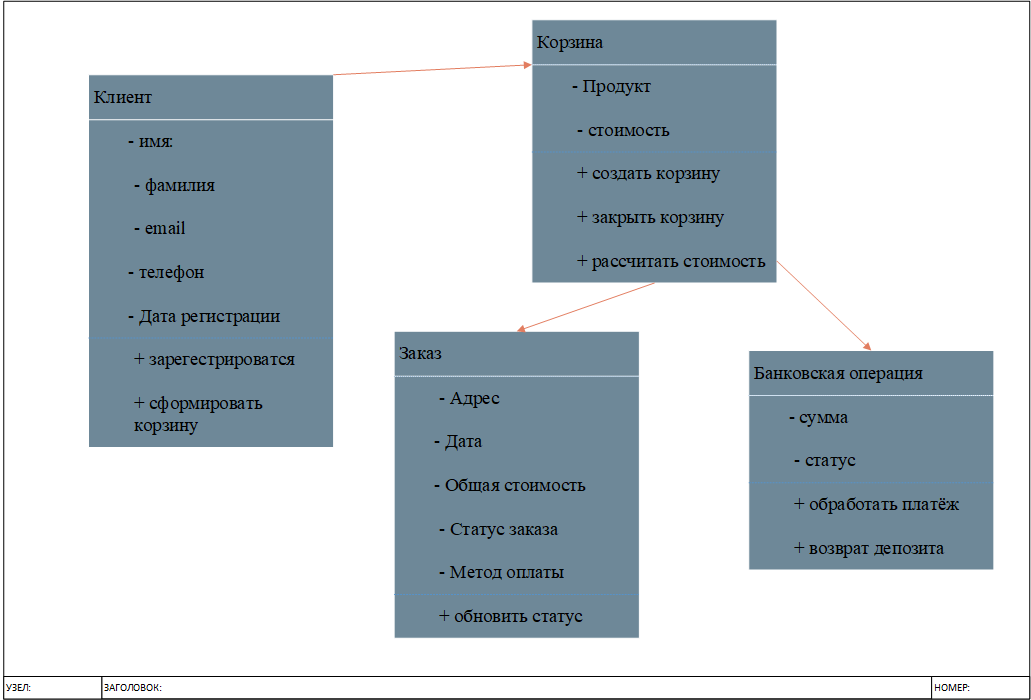


Рис. 6 UML – Диаграмма классов

**1.6.5 Бизнес – процессы**

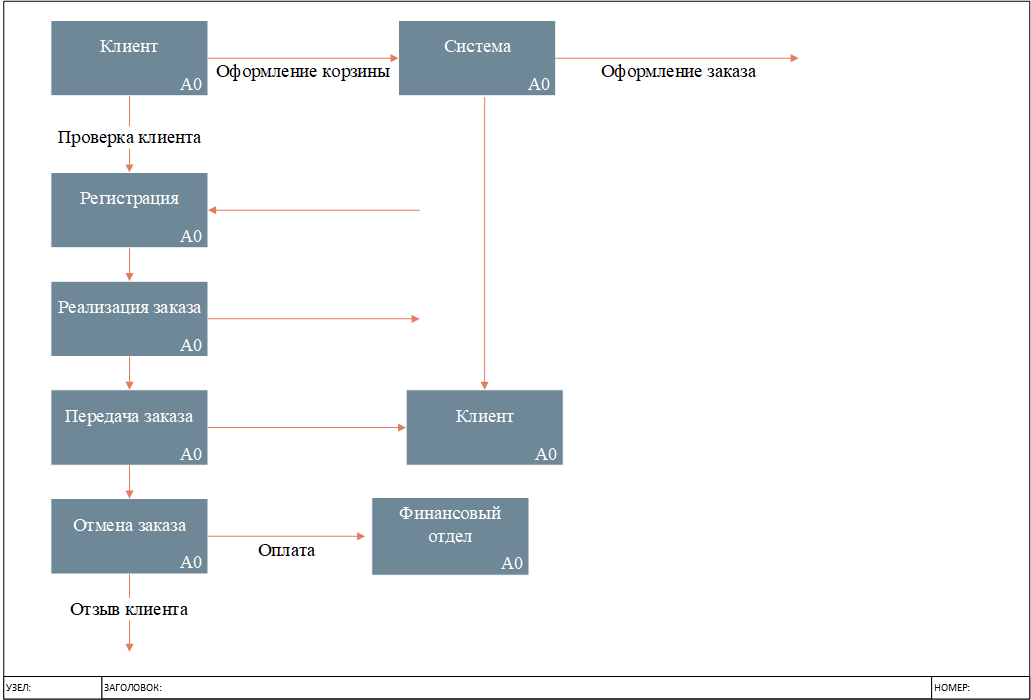


Рис. 7 Бизнес – процессы

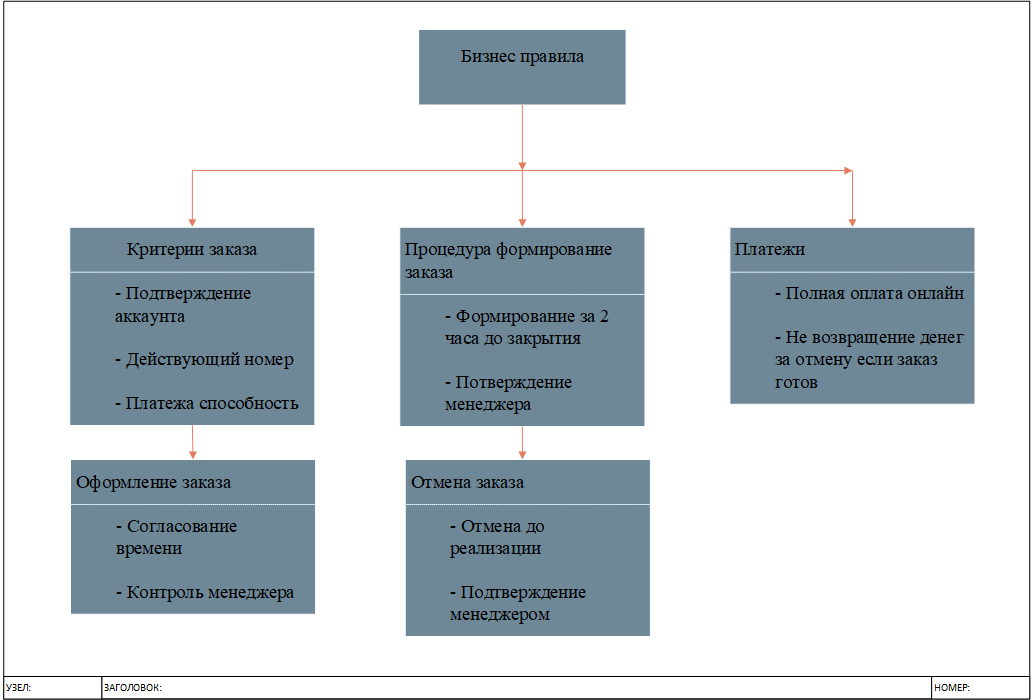


Рис. 8 Определение бизнес правил

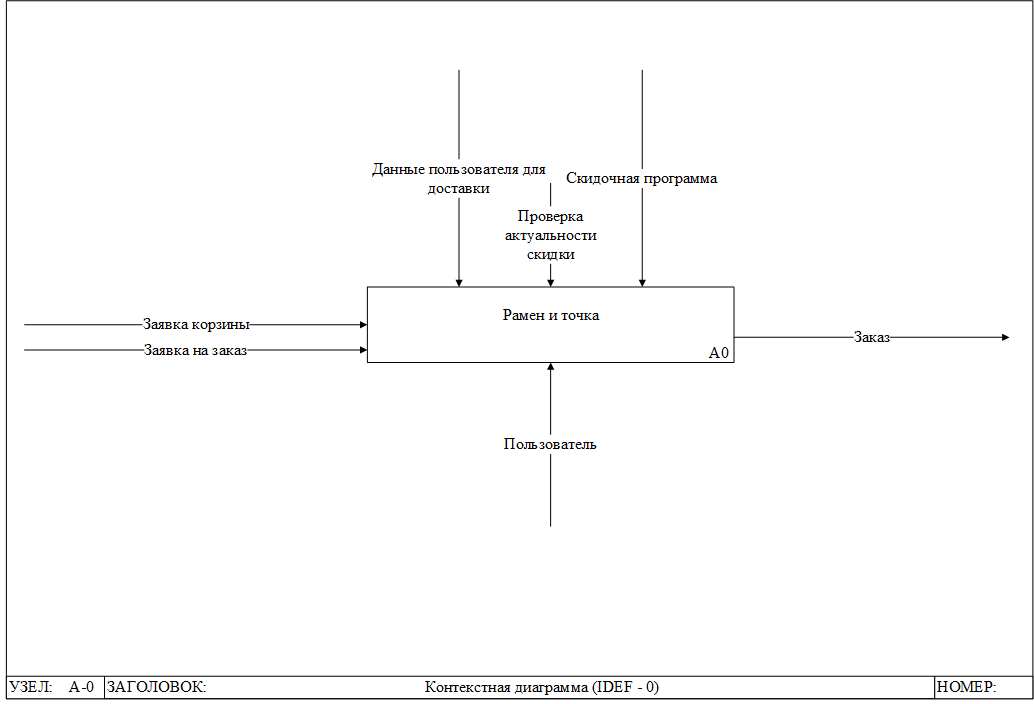


Рис. 9 Контекстная диаграмма IDEF-0

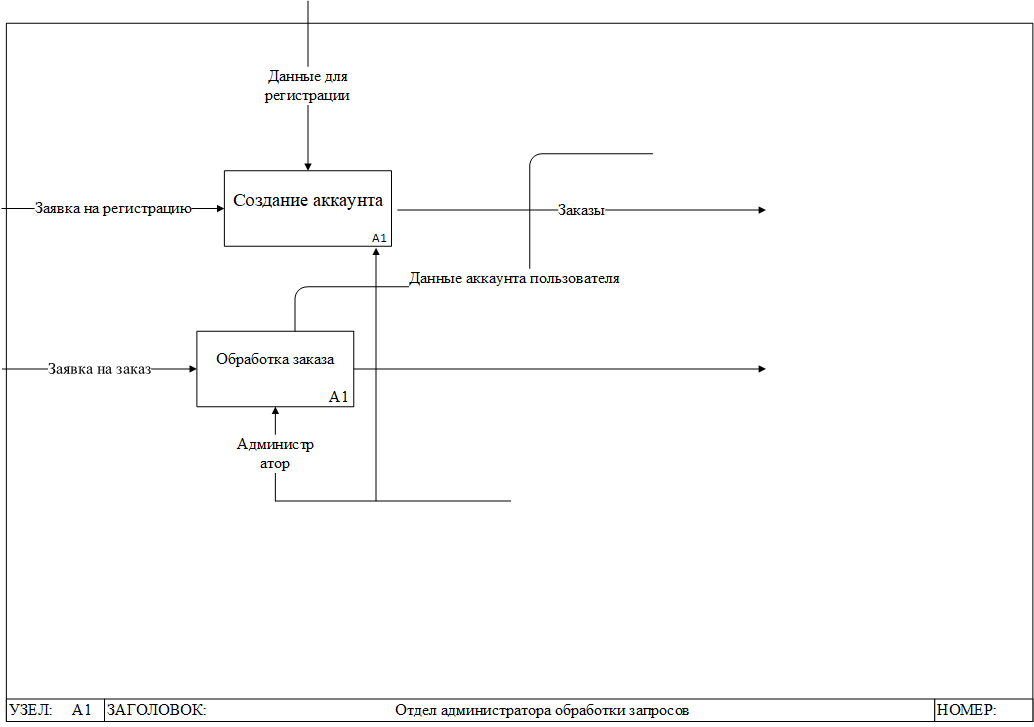


Рис. 10 Отдел администратора обработки запроса IDEF-1

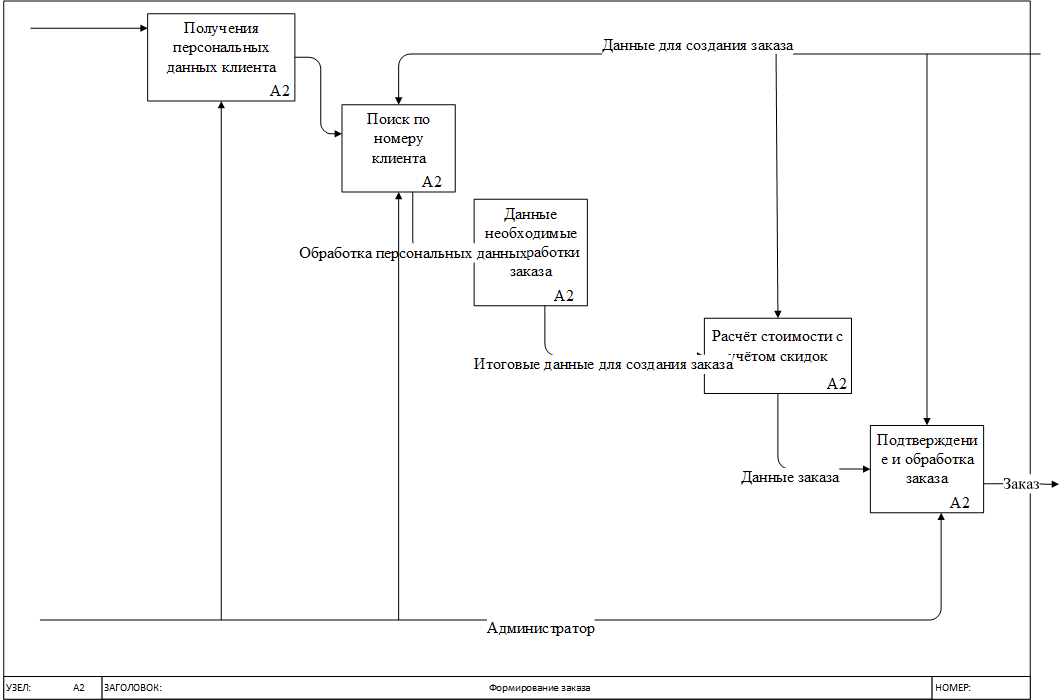


Рис. 11 Формирование заказа IDEF-2

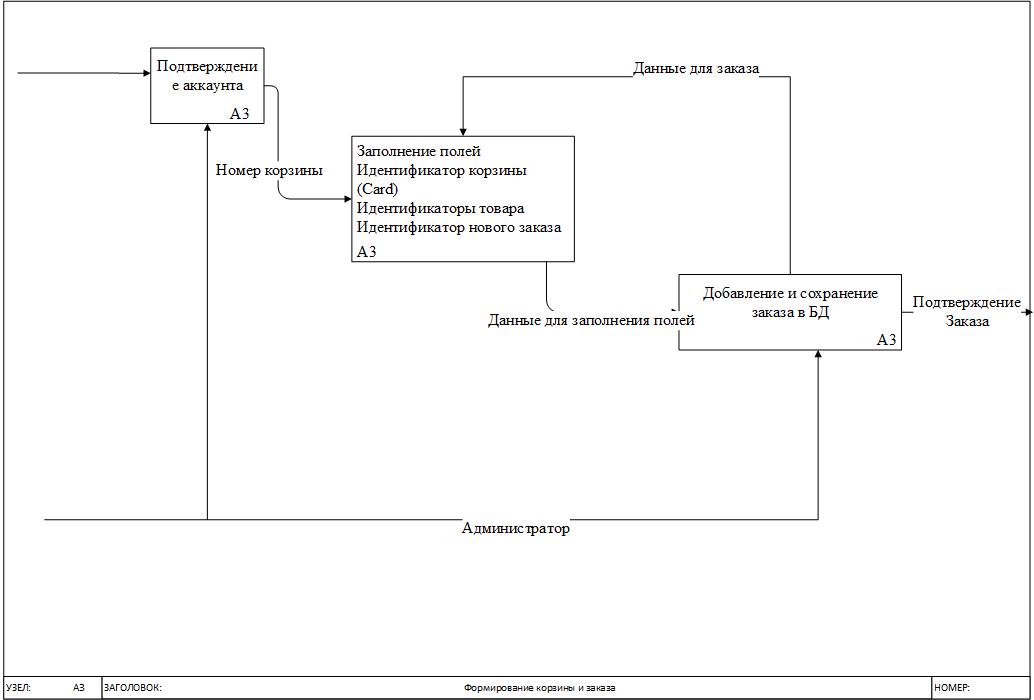


Рис. 12 Формирование корзины и заказа IDEF-3

# **ГЛАВА 2. РАЗРАБОТКА БАЗЫ ДАННЫХ**

**2.1 Разработка логической модели данных**

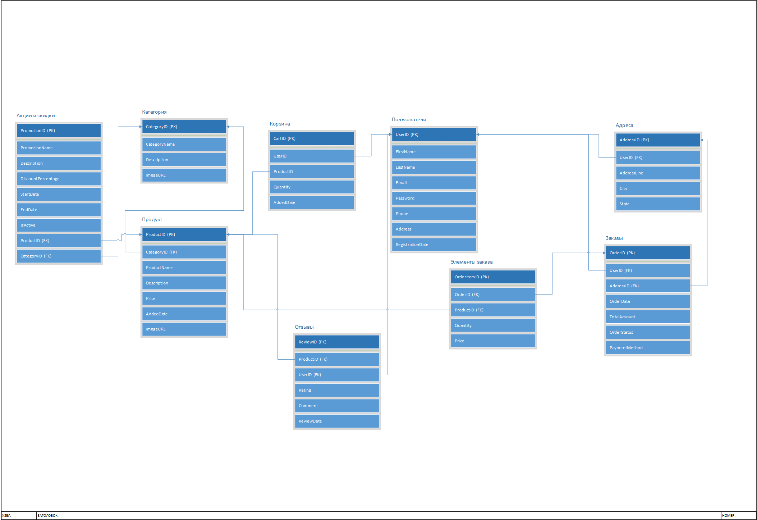


Рис. 13 Логическая модель данных

**2.2 Добавление таблиц к базе данных**

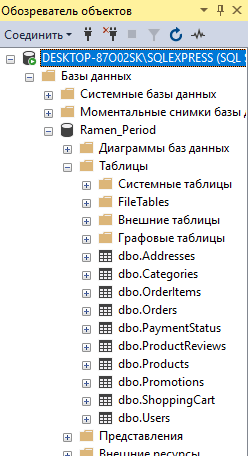


Рис. 14 Добавление таблиц к базе данных

* 1. **Связи на диаграмме базе данных**

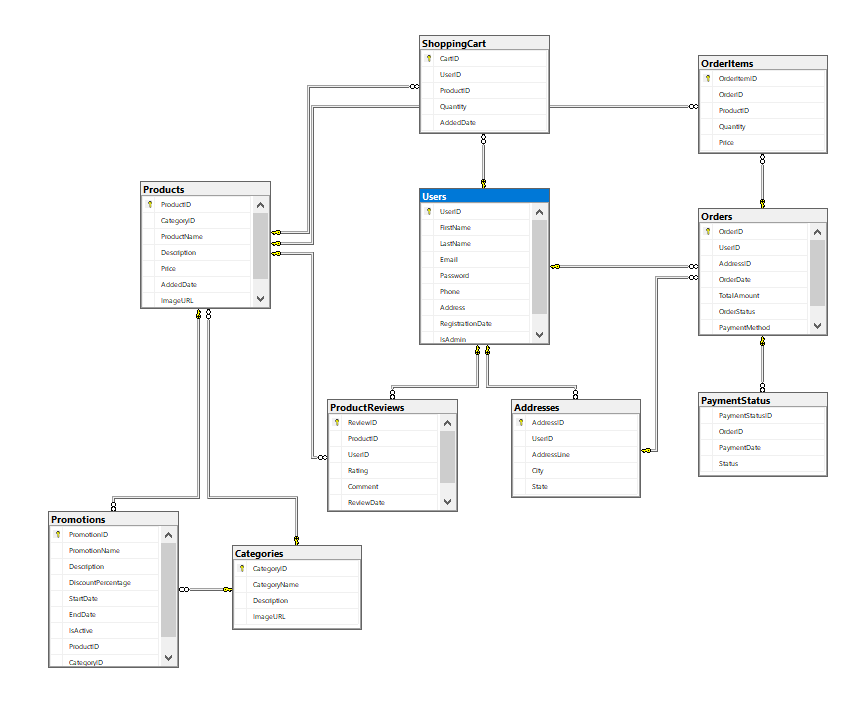


Рис. 15 Связи в базе данных

**2.4 Определение ограничений налагаемые на данные**

- Текст ограничен до 255 символов в запись.  
- Дата ограничена от 0L до 3155378975999999999L (от 01.01.0001 0:00:00 до 31.12.9999 23:59:59)

- Числа от -2147483648 до 2147483647

- Не целые числа от -3,4028235E+38 до 3,4028235E+38

- Доступные значения в Orders.OrderStatus: Сформирован, Выполняется, Отменён, Передан в доставку, Доставлен

- Доступные значение в Orders.PaymentStatus: Наличные, Картой, Онлайн.

- Доступные значение в PaymentStatus.Status: Ожидает, Отменён, Оплачен

**2.5 Разработка физической структуры данных**

--Categories +

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (1, 'Классический рамен', 'Традиционные японские рецепты рамена.');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (2, 'Острый рамен', 'Рамен с добавлением острых специй и соусов.');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (3, 'Вегетарианский рамен', 'Рамен без мяса, с использованием овощей и тофу.');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (4, 'Напитки', 'Прохладительные и горячие напитки.');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (5, 'Дополнительные ингредиенты', 'Дополнительные топпинги к рамену');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (6, 'Рамен быстрого приготовления', 'Быстро и вкусно!');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (7, 'Детское меню', 'Специально для детей');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (8, 'Десерты', 'Сладкое к рамену');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (9, 'Сезонные предложения', 'Ограниченные по времени блюда.');

INSERT INTO Categories (CategoryID, CategoryName, [Description]) VALUES (10, 'Комбо-наборы', 'Рамен + напиток + закуска по выгодной цене.');

--Products +

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (1, 1, 'Рамен Тонкоцу', 'Насыщенный свиной бульон, лапша, свинина чашу, яйцо, зеленый лук.', 450.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (2, 2, 'Каракай Рамен', 'Острый куриный бульон, лапша, курица, перец чили, кунжут.', 500.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (3, 3, 'Овощной Рамен', 'Овощной бульон, лапша, тофу, грибы шиитаке, морковь, шпинат.', 400.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (4, 4, 'Coca-Cola', 'Охлажденная Coca-Cola.', 100.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (5, 5, 'Яйцо Аджитама', 'Маринованное яйцо.', 50.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (6, 6, 'Рамен со вкусом курицы', 'Просто добавь воды!', 150.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (7, 1, 'Шио Рамен', 'Бульон на основе морской соли, лапша, курица, нори, зеленый лук.', 420.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (8, 8, 'Моти', 'Традиционный японский десерт из рисового теста.', 200.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (9, 9, 'Рамен с креветками (зимний)', 'Креветочный бульон, лапша, креветки, овощи.', 550.00);

INSERT INTO Products (ProductID, CategoryID, ProductName, [Description], Price) VALUES (10, 10, 'Комбо "Классика"', 'Рамен Тонкоцу, Coca-Cola, Яйцо Аджитама.', 550.00);

--Users +

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone) VALUES (1, 'Иван', 'Иванов', 'ivan@example.com', 'hashed\_[Password]\_1', '+79001234567');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone) VALUES (2, 'Мария', 'Петрова', 'maria@example.com', 'hashed\_[Password]\_2', '+79123456789');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password]) VALUES (3, 'Алексей', 'Сидоров', 'alex@example.com', 'hashed\_[Password]\_3');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone, Address) VALUES (4, 'Елена', 'Смирнова', 'elena@example.com', 'hashed\_[Password]\_4', '+79234567890', 'ул. Ленина, 10');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password]) VALUES (5, 'Дмитрий', 'Козлов', 'dmitry@example.com', 'hashed\_[Password]\_5');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone) VALUES (6, 'Ольга', 'Волкова', 'olga@example.com', 'hashed\_[Password]\_6', '+79345678901');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password]) VALUES (7, 'Сергей', 'Морозов', 'sergey@example.com', 'hashed\_[Password]\_7');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone, Address) VALUES (8, 'Наталья', 'Лебедева', 'natalia@example.com', 'hashed\_[Password]\_8', '+79456789012', 'пр. Мира, 25');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password]) VALUES (9, 'Андрей', 'Павлов', 'andrey@example.com', 'hashed\_[Password]\_9');

INSERT INTO Users (UserID, FirstName, LastName, Email, [Password], Phone) VALUES (10, 'Юлия', 'Николаева', 'julia@example.com', 'hashed\_[Password]\_10', '+79567890123');

--addresses +

INSERT INTO Addresses (AddressID, UserID, AddressLine, City, State) VALUES (1, 1, 'ул. Пушкина, 5', 'Москва', 'Московская область');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City) VALUES (2, 1, 'пр. Гагарина, 12', 'Санкт-Петербург');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City) VALUES (3, 2, 'ул. Ленина, 20', 'Екатеринбург');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City, State) VALUES (4, 3, 'ул. Мира, 15', 'Новосибирск', 'Новосибирская область');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City) VALUES (5, 4, 'ул. Строителей, 8', 'Казань');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City, State) VALUES (6, 5, 'ул. Советская, 3', 'Нижний Новгород', 'Нижегородская область');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City) VALUES (7, 6, 'ул. Кирова, 10', 'Челябинск');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City, State) VALUES (8, 7, 'ул. 8 Марта, 7', 'Самара', 'Самарская область');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City) VALUES (9, 8, 'ул. Карла Маркса, 1', 'Омск');

INSERT INTO Addresses (AddressID, UserID, AddressLine, City, State) VALUES (10, 9, 'ул. Первомайская, 12', 'Ростов-на-Дону', 'Ростовская область');

--Orders +

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (1, 1, 1, 900.00, 'Доставлен', 'Картой');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (2, 2, 3, 450.00, 'Выполняется', 'Наличные');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (3, 3, 4, 1000.00, 'Сформирован', 'Онлайн');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (4, 4, 5, 600.00, 'Передан в доставку', 'Картой');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (5, 5, 6, 300.00, 'Отменён', 'Наличные');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (6, 6, 7, 750.00, 'Доставлен', 'Онлайн');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (7, 7, 8, 500.00, 'Выполняется', 'Картой');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (8, 8, 9, 1200.00, 'Сформирован', 'Наличные');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (9, 9, 10, 850.00, 'Передан в доставку', 'Онлайн');

INSERT INTO Orders (OrderID, UserID, AddressID, TotalAmount, OrderStatus, PaymentMethod) VALUES (10, 10, 2, 400.00, 'Отменён', 'Картой');

--OrderItem +

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (1, 1, 1, 2, 450.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (2, 1, 4, 1, 100.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (3, 2, 2, 1, 500.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (4, 3, 3, 2, 400.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (5, 3, 5, 4, 50.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (6, 4, 1, 1, 450.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (7, 4, 4, 1, 100.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (8, 5, 2, 1, 500.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (9, 6, 3, 1, 400.00);

INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity, Price) VALUES (10, 6, 5, 2, 50.00);

--shopcart +

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (1, 1, 1, 1);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (2, 1, 4, 2);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (3, 2, 2, 1);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (4, 3, 3, 3);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (5, 4, 1, 2);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (6, 5, 2, 1);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (7, 6, 3, 1);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (8, 7, 1, 1);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (9, 8, 2, 2);

INSERT INTO ShoppingCart (CartID, UserID, ProductID, Quantity) VALUES (10, 9, 3, 1);

--comments +

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (1, 1, 1, 5, 'Отличный рамен!');

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (2, 1, 2, 4, 'Вкусно, но немного дорого.');

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating) VALUES (3, 2, 3, 4);

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (4, 3, 4, 5, 'Очень понравился вегетарианский вариант!');

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating) VALUES (5, 1, 5, 2);

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (6, 2, 6, 4, 'Острый, как я люблю!');

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating) VALUES (7, 3, 7, 5);

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (8, 1, 8, 3, 'Неплохо, но можно и лучше.');

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating) VALUES (9, 2, 9, 3);

INSERT INTO ProductReviews (ReviewID, ProductID, UserID, Rating, Comment) VALUES (10, 3, 10, 4, 'Рекомендую!');

--discount +

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, ProductID) VALUES (1, 'Скидка на Тонкоцу', 'Скидка 10% на рамен Тонкоцу.', 10.00, 1);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, CategoryID) VALUES (2, 'Счастливые часы', 'Скидка 15% на все напитки с 14:00 до 16:00.', 15.00, 4);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage) VALUES (3, 'Комбо по цене рамена', 'При покупке рамена - напиток в подарок.', 100.00);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, ProductID) VALUES (4, 'Рамен месяца', 'Скидка на рамен месяца.', 25.00, 2);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, CategoryID) VALUES (5, 'Вегетарианский вторник', 'Скидка 20% на все вегетарианские блюда по вторникам.', 20.00, 3);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, ProductID) VALUES (6, 'Счастливый обед', 'Скидка 30% с 12:00 до 14:00 на всё меню.', 30.00, 3);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, CategoryID) VALUES (7, 'Ко дню рождения!', 'Скидка 15% в день рождения и 3 дня после.', 15.00, 7);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, ProductID) VALUES (8, 'Попробуй первым!', 'Скидка 15% на новинки.', 15.00, 8);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, CategoryID) VALUES (9, 'Приведи друга - получи скидку!', 'Приведи друга - получи скидку 10% на свой заказ.', 10.00, 9);

INSERT INTO Promotions (PromotionID, PromotionName, [Description], DiscountPercentage, ProductID) VALUES (10, 'Закажи на 2000 - получи десерт в подарок!', 'Закажи на сумму от 2000 - получи десерт в подарок!', 50.00, 9);

--payments +

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (1, 1, 'Оплачен');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (2, 2, 'Ожидает');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (3, 3, 'Оплачен');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (4, 4, 'Ожидает');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (5, 5, 'Отменён');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (6, 6, 'Оплачен');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (7, 7, 'Ожидает');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (8, 8, 'Ожидает');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (9, 9, 'Оплачен');

INSERT INTO PaymentStatus(PaymentStatusID, OrderID, [Status]) VALUES (10, 10, 'Отменён');

**2.6 Создание БД и таблиц**

-- Таблица: Категории товаров (рамен, добавки, напитки и т.д.)

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY NOT NULL,

CategoryName VARCHAR(MAX) NOT NULL,

Description VARCHAR(MAX),

ImageURL VARCHAR(MAX) DEFAULT 'https://sun9-10.userapi.com/impg/IOxWhTf5vHdaErwvUHz6LpVBtdbQXH\_eUBHG-w/JrxHi27cfr8.jpg?size=320x320&quality=95&sign=cbbcb855a0aa5040958e0957f92f477b&type=album'

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY NOT NULL,

CategoryID INT NOT NULL,

ProductName VARCHAR(MAX) NOT NULL,

Description VARCHAR(MAX),

Price DECIMAL(10, 2) NOT NULL,

AddedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

ImageURL VARCHAR(MAX) DEFAULT 'https://sun9-10.userapi.com/impg/IOxWhTf5vHdaErwvUHz6LpVBtdbQXH\_eUBHG-w/JrxHi27cfr8.jpg?size=320x320&quality=95&sign=cbbcb855a0aa5040958e0957f92f477b&type=album', -- Путь к изображению товара

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

-- Таблица: Пользователи (покупатели)

CREATE TABLE [Users] (

UserID INT PRIMARY KEY NOT NULL,

FirstName VARCHAR(MAX) NOT NULL,

LastName VARCHAR(MAX) NOT NULL,

Email VARCHAR(MAX) NOT NULL,

[Password] VARCHAR(255) NOT NULL,

Phone VARCHAR(20),

Address VARCHAR(MAX),

RegistrationDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

IsAdmin BIT NOT NULL DEFAULT 0

);

-- Таблица: Адреса доставки (для каждого пользователя может быть несколько адресов)

CREATE TABLE Addresses (

AddressID INT PRIMARY KEY NOT NULL,

UserID INT NOT NULL,

AddressLine VARCHAR(MAX) NOT NULL, --УЛИЦА

City VARCHAR(MAX) NOT NULL, --Город

[State] VARCHAR(MAX), --Область

FOREIGN KEY (UserID) REFERENCES [Users](UserID)

);

-- Таблица: Заказы

CREATE TABLE Orders (

OrderID INT PRIMARY KEY NOT NULL,

UserID INT NOT NULL,

AddressID INT,

OrderDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

TotalAmount DECIMAL(10, 2) NOT NULL,

OrderStatus VARCHAR(MAX) NOT NULL DEFAULT 'Сформирован' CHECK(OrderStatus = 'Сформирован' OR OrderStatus = 'Выполняется' OR OrderStatus = 'Отменён' OR OrderStatus = 'Передан в доставку' OR OrderStatus = 'Доставлен' ),

PaymentMethod VARCHAR(MAX) NOT NULL DEFAULT 'Наличные' CHECK(PaymentMethod = 'Наличные' OR PaymentMethod = 'Картой' OR PaymentMethod = 'Онлайн'),

FOREIGN KEY (UserID) REFERENCES [Users](UserID),

FOREIGN KEY (AddressID) REFERENCES Addresses(AddressID)

);

-- Таблица: Позиции заказа (какие товары в каком количестве входят в заказ)

CREATE TABLE OrderItems (

OrderItemID INT PRIMARY KEY NOT NULL,

OrderID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL,

Price DECIMAL(10, 2) NOT NULL, -- Цена товара на момент заказа

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

-- Таблица: Корзина покупок (временное хранение товаров перед оформлением заказа)

CREATE TABLE ShoppingCart (

CartID INT PRIMARY KEY NOT NULL,

UserID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL,

AddedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (UserID) REFERENCES [Users](UserID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

-- Таблица: Отзывы о товарах

CREATE TABLE ProductReviews (

ReviewID INT PRIMARY KEY NOT NULL,

ProductID INT NOT NULL,

UserID INT NOT NULL,

Rating INT NOT NULL CHECK (Rating >= 1 AND Rating <= 5), -- Оценка от 1 до 5

[Comment] VARCHAR(MAX),

ReviewDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (ProductID) REFERENCES Products(ProductID),

FOREIGN KEY (UserID) REFERENCES [Users](UserID)

);

-- Таблица: Акции и скидки

CREATE TABLE Promotions (

PromotionID INT PRIMARY KEY NOT NULL,

PromotionName VARCHAR(MAX) NOT NULL,

Description VARCHAR(MAX),

DiscountPercentage DECIMAL(5, 2) NOT NULL,

StartDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

EndDate DATETIME DEFAULT CURRENT\_TIMESTAMP + 7, --По умолчанию акция на 7 дней

IsActive BIT DEFAULT 1,

ProductID INT, -- Ссылка на конкретный товар (если акция только на него)

CategoryID INT, -- Ссылка на категорию (если акция на категорию)

FOREIGN KEY (ProductID) REFERENCES Products(ProductID),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

--Таблица: Статус платежей

CREATE TABLE PaymentStatus(

PaymentStatusID INT NOT NULL,

OrderID INT NOT NULL,

PaymentDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

[Status] VARCHAR(16) NOT NULL DEFAULT 'Ожидает' CHECK([Status] = 'Ожидает' OR [Status] = 'Отменён' OR [Status] = 'Оплачен'),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

)

**2.7 Данные в таблицах**

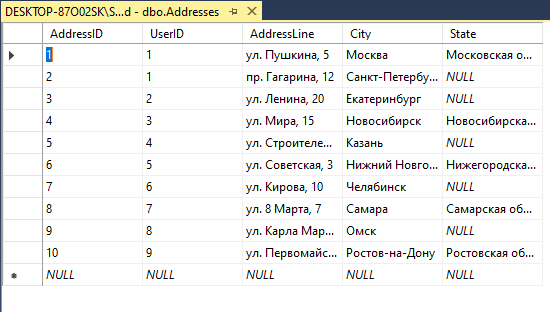


Рис. 16 Таблица «Адреса»

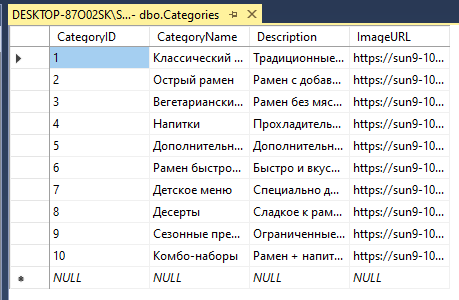


Рис. 17 Таблица «Категории»

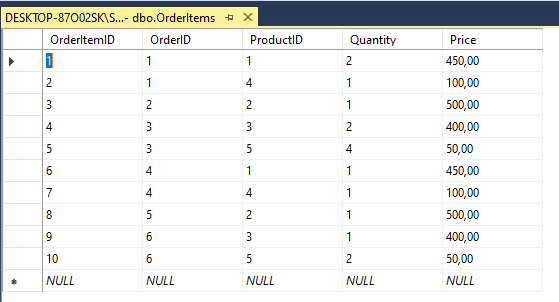


Рис. 18 Таблица «Элементы заказа»

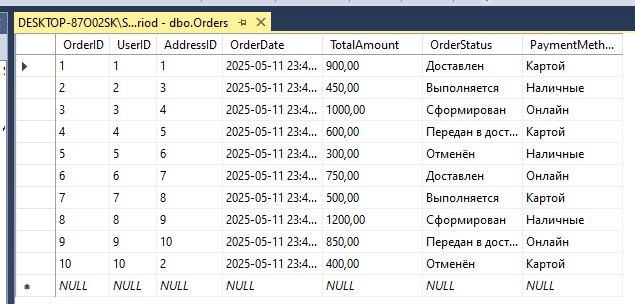


Рис. 19 Таблица «Заказы»

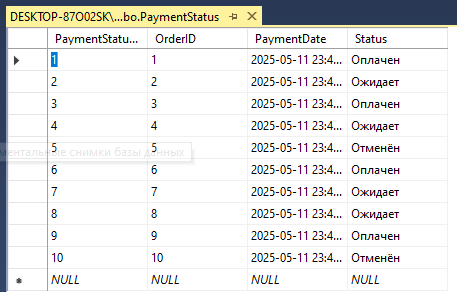


Рис. 20 Таблица «Платежи»

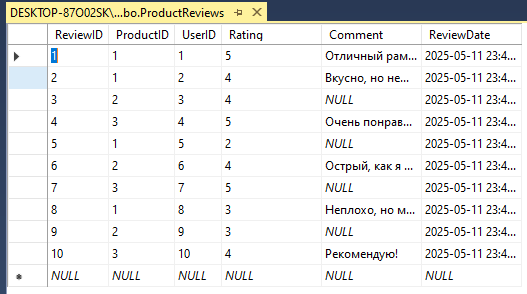


Рис. 21 Таблица «Отзывы»

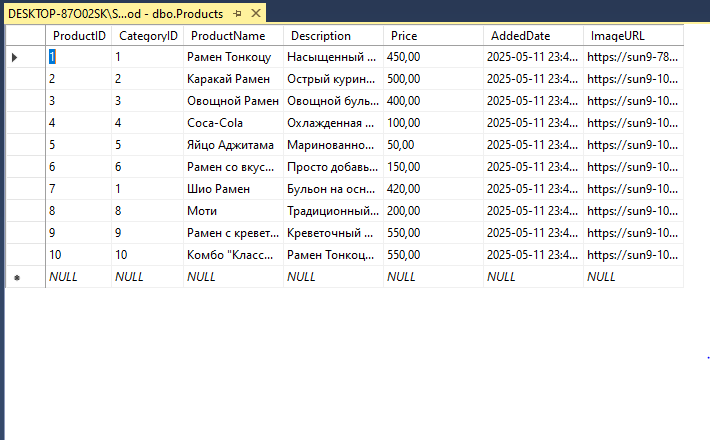


Рис. 22 Таблица «Продукты»

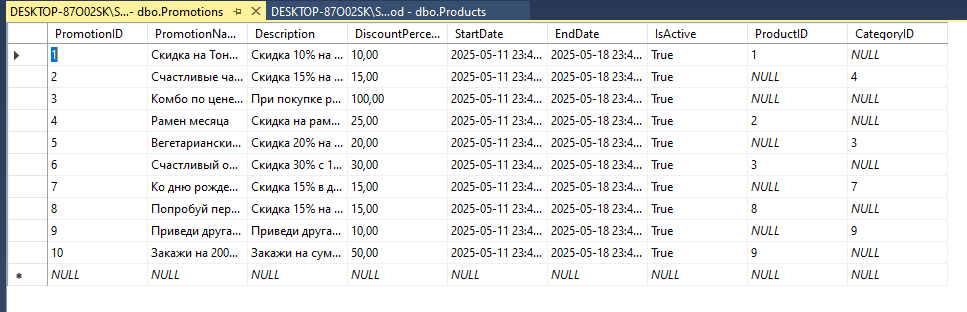


Рис. 23 Таблица «Скидки и акции»

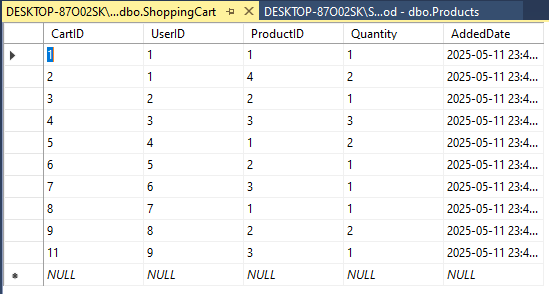


Рис. 24 Таблица «Корзина»

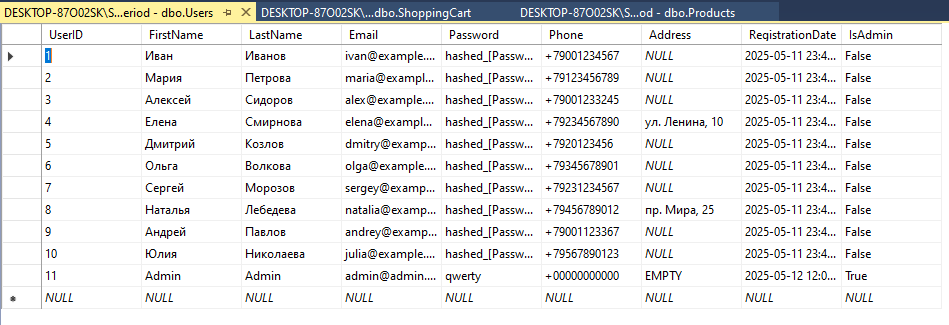


Рис. 25 Таблица «Пользователи»

**2.8 Создание запросов**

1. Общий средний рейтинг для всех товаров:

SELECT AVG(Rating) AS AverageRating

FROM ProductReviews;

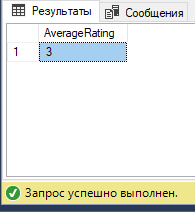


Рис. 26 Запрос на средний рейтинг

2. Средний рейтинг для каждого товара:

SELECT ProductID, AVG(Rating) AS AverageRating

FROM ProductReviews

GROUP BY ProductID;

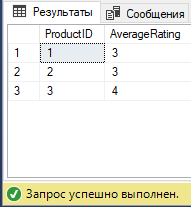


Рис. 27 Запрос на рейтинг каждого товара

3. Средний рейтинг для каждого товара, отсортированный по убыванию рейтинга:

SELECT ProductID, AVG(Rating) AS AverageRating

FROM ProductReviews

GROUP BY ProductID

ORDER BY AverageRating DESC;

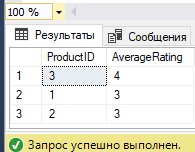


Рис. 28 Запрос рейтинг для каждого товара отсортированного по убыванию

4. Общее количество отзывов и средний рейтинг для каждого товара:

SELECT ProductID, COUNT(\*) AS NumberOfReviews, AVG(Rating) AS AverageRating

FROM ProductReviews

GROUP BY ProductID;

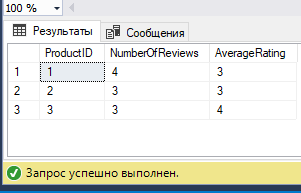


Рис. 29 Общее кол.во отзывов и средний рейтинг

1. Только товары с определенным минимальным количеством отзывов:

SELECT ProductID, COUNT(\*) AS NumberOfReviews, AVG(Rating) AS AverageRating

FROM ProductReviews

GROUP BY ProductID

HAVING COUNT(\*) >= 1;

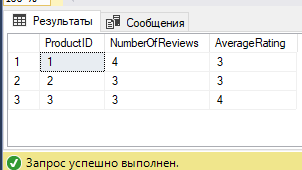


Рис. 30 Товар с опред. Минимальным количеством отзывов

**ГЛАВА 3. РАЗРАБОТКА ПРИЛОЖЕНИЯ**

**3.1 Разработка пользовательского интерфейса**

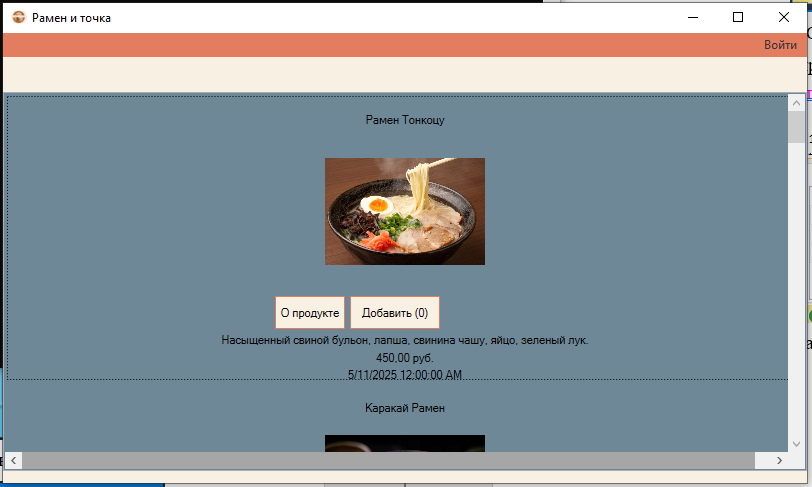


Рис. 31 Главная форма

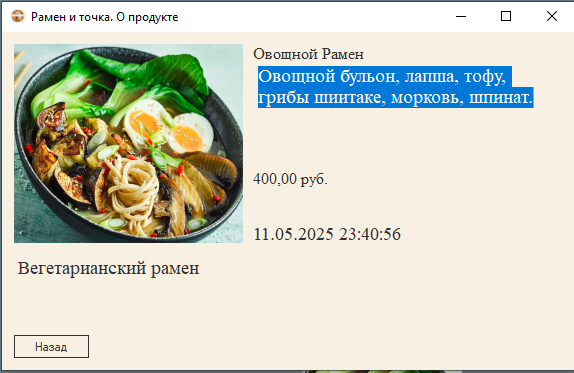


Рис. 32 Форма «О продукте»

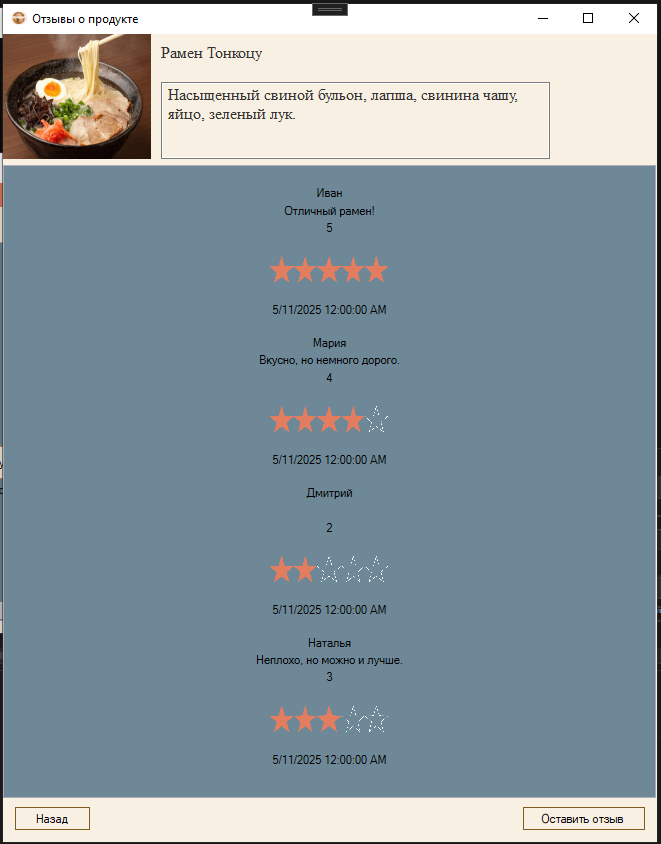


Рис. 33 Форма «Отзыв о товаре»

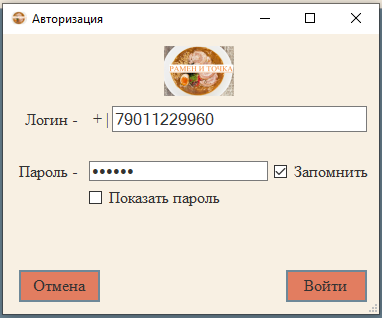


Рис. 34 Форма «Авторизация»

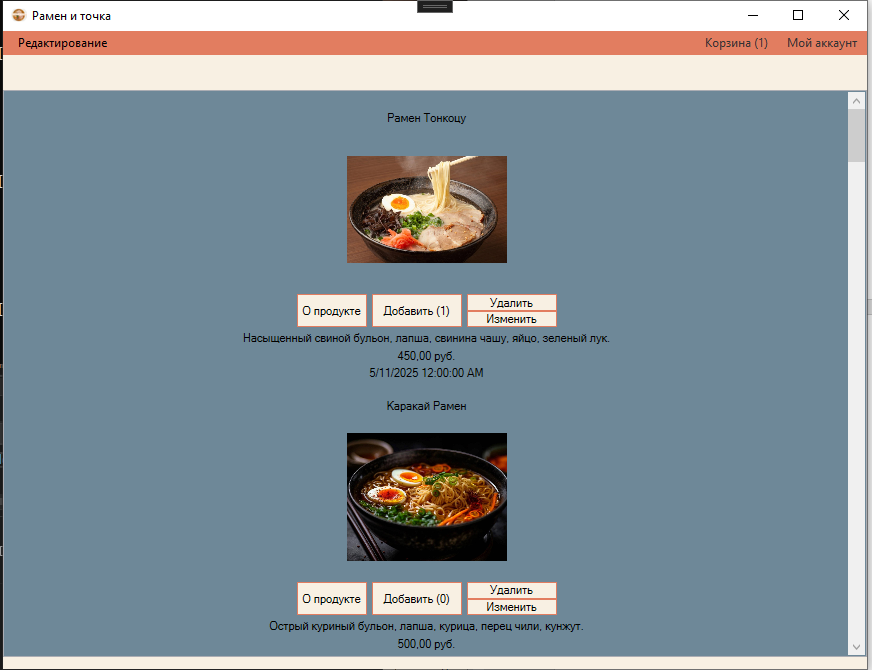


Рис. 35 Главная форма после авторизации

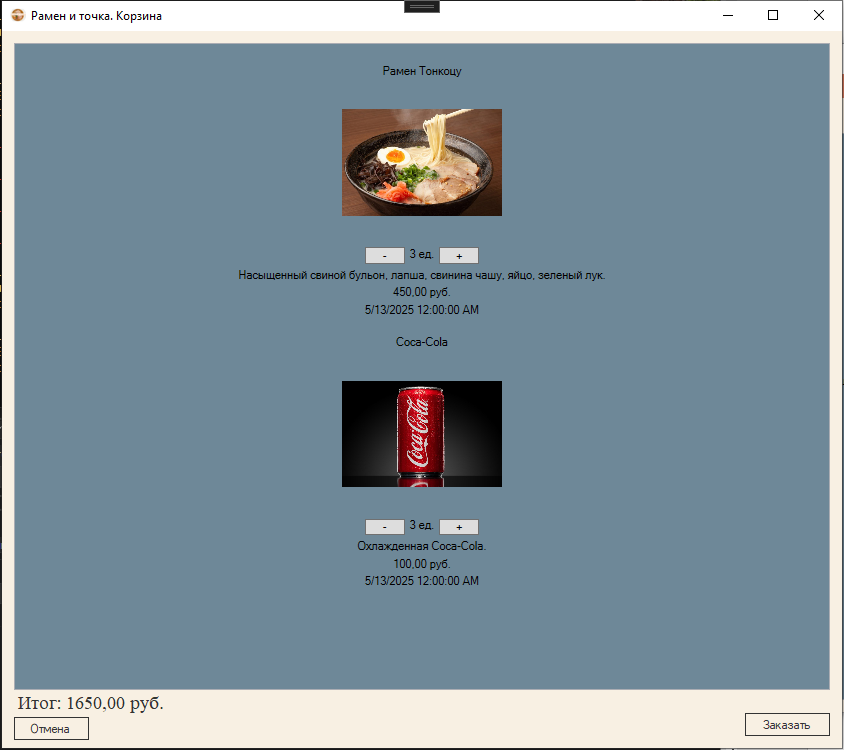


Рис. 36 Форма «Корзина»

**3.2 Разработка библиотек класса**

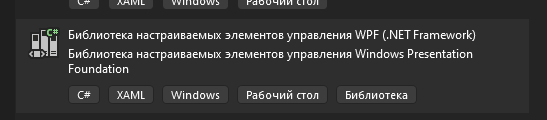


Рис. 37 Создание пользовательского элемента WPF

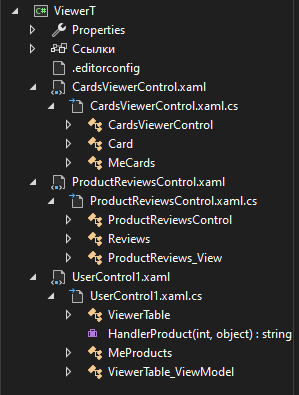


Рис. 38 Создание элементов

**3.3 Листинг программы**

Главный код формы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using Ramen\_period.Dialogs;

namespace Ramen\_period

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

DialogResult = DialogResult.Retry;

BindL<MainForm>.Set(this);

}

private void MainForm\_Load(object sender, EventArgs e)

{

UpdateLentaProducts();

UpdateState();

UpdateCardsCount();

var tms = new System.Windows.Forms.Timer();

tms.Tick += Timer\_Handler;

tms.Interval = 5000;

tms.Start();

}

private string Last = "";

private void Timer\_Handler(object sender,EventArgs e)

{

Last = cardsToolStripMenuItem.Text;

UpdateState();

UpdateCardsCount();

if(cardsToolStripMenuItem.Text != Last)

{

UpdateLentaProducts();

}

}

/// <summary>

/// Событие для кнопки "Войти"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void AuthToolStripMenuItem\_Click(object sender, EventArgs e)

{

Dialogs.AuthDialogForm authF = new Dialogs.AuthDialogForm();

authF.Owner = this;

authF.DialogResult = DialogResult.None;

authF.ShowDialog();

UpdateState();

UpdateCardsCount();

}

public void UpdateState()

{

if (Program.IsAuth())

{

Debugger.Log(11, PATHCODE.WARNING, "IsAuth True", typeof(MainForm));

AuthToolStripMenuItem.Text = "Мой аккаунт";

//Очистка событий (нужно чтобы избежать повторности вызыва)

AuthToolStripMenuItem.Click -= AuthToolStripMenuItem\_Click;

AuthToolStripMenuItem.Click -= DeAuthToolStripMenuItem\_Click;

//Добавление чистого события

AuthToolStripMenuItem.Click += DeAuthToolStripMenuItem\_Click;

cardsToolStripMenuItem.Visible = true;

foreach (var v in ((hostElementTable.Child as ViewerT.ViewerTable).DataContext as ViewerT.ViewerTable\_ViewModel).MeProducts)

{

v.DelVisible = System.Windows.Visibility.Visible;

}

if (Program.IsAdmin())

{

EditToolStripMenuItem.Visible = true;

foreach (var v in ((hostElementTable.Child as ViewerT.ViewerTable).DataContext as ViewerT.ViewerTable\_ViewModel).MeProducts)

{

v.EditVisible = System.Windows.Visibility.Visible;

}

}

else

{

foreach (var v in ((hostElementTable.Child as ViewerT.ViewerTable).DataContext as ViewerT.ViewerTable\_ViewModel).MeProducts)

{

v.EditVisible = System.Windows.Visibility.Hidden;

}

EditToolStripMenuItem.Visible = false;

}

}

else

{

AuthToolStripMenuItem.Text = "Войти";

AuthToolStripMenuItem.Click -= AuthToolStripMenuItem\_Click;

AuthToolStripMenuItem.Click -= DeAuthToolStripMenuItem\_Click;

//Добавление чистого события

AuthToolStripMenuItem.Click += AuthToolStripMenuItem\_Click;

foreach (var v in ((hostElementTable.Child as ViewerT.ViewerTable).DataContext as ViewerT.ViewerTable\_ViewModel).MeProducts)

{

v.DelVisible = System.Windows.Visibility.Hidden;

v.EditVisible = System.Windows.Visibility.Hidden;

((hostElementTable.Child as ViewerT.ViewerTable)).UpdateDefaultStyle();

((hostElementTable.Child as ViewerT.ViewerTable)).UpdateLayout();

}

cardsToolStripMenuItem.Visible = false;

EditToolStripMenuItem.Visible = false;

}

}

private void UpdateLentaProducts()

{

while (true)

{

try

{

Program.SQL.CreateCommand();

break;

}

catch

{

continue;

}

}

List<ViewerT.MeProducts> prods = new List<ViewerT.MeProducts>();

var com = Program.SQL.CreateCommand();

com.CommandText = "SELECT \* FROM [Products]";

var reader = com.ExecuteReader();

DataTable tab = new DataTable();

tab.Load(reader);

reader.Close();

for (int i = 0; i < tab.Rows.Count; i++)

{

ViewerT.MeProducts prod = new ViewerT.MeProducts();

prod.IdProduct = int.Parse(tab.Rows[i].ItemArray[0].ToString());

prod.ProductName = tab.Rows[i].ItemArray[2].ToString();

prod.Description = tab.Rows[i].ItemArray[3].ToString();

prod.Price = tab.Rows[i].ItemArray[4].ToString();

prod.AddedDate = DateTime.Parse(tab.Rows[i].ItemArray[5].ToString());

//pre-load for shop cards

var check\_com = Program.SQL.CreateCommand();

check\_com.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [ProductID] = \'{tab.Rows[i].ItemArray[0]}\' AND [UserID] = \'{Program.GetUID()}\'";

using (var rd = check\_com.ExecuteReader())

{

if (rd.HasRows)

{

DataTable table = new DataTable();

table.Load(rd);

if (table.Rows.Count > 0)

prod.CountAdded = table.Rows[0].Field<int>("Quantity").ToString();

}

}

prod.Set\_Image\_Url(tab.Rows[i].ItemArray[6].ToString());

prods.Add(prod);

}

var ACT\_ABOUT = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

var obj = tab.Rows.Cast<DataRow>().Where(x => x.ItemArray[0].ToString() == IDP.ToString()).FirstOrDefault();

Dialogs.AboutProductForm f\_about = new AboutProductForm(IDP);

f\_about.Owner = this;

// f\_about.Show();

Dialogs.ProductReviewsForm rev = new ProductReviewsForm(IDP);

rev.Show();

return string.Empty;

});

var ACT\_ADD = new ViewerT.HandlerProduct((IDP,\_sender) =>

{

if (Program.IsAuth())

{

var check\_com = Program.SQL.CreateCommand();

bool create = true;

check\_com.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [ProductID] = \'{IDP}\' AND [UserID] = \'{Program.GetUID()}\'";

using(var rd = check\_com.ExecuteReader())

{

if(rd.HasRows)

{

create = false;

}

}

if(create)

{

var create\_new\_card = Program.SQL.CreateCommand();

int newIDCard = Program.GetMaxID("ShoppingCart", "CartID")+1;

create\_new\_card.CommandText = $"INSERT INTO [ShoppingCart] (CartID, UserID, ProductID, Quantity) VALUES({newIDCard},{Program.GetUID()},{IDP},1)";

create\_new\_card.ExecuteNonQuery();

MessageBox.Show(this, "Товар добавлен в корзину", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

if (\_sender is System.Windows.Controls.MenuItem)

{

UpdateLentaProducts();

}

return $"Добавить (1)";

}

else

{

int quant = 1;

using(var rd = check\_com.ExecuteReader())

{

var tb = new DataTable();

tb.Load(rd);

quant = tb.Rows[0].Field<int>("Quantity");

}

var com\_update = Program.SQL.CreateCommand();

quant += 1;

com\_update.CommandText = $"UPDATE ShoppingCart SET [Quantity] = {quant} WHERE [ProductID] = '{IDP}' AND [UserID] = '{Program.GetUID()}'";

com\_update.ExecuteNonQuery();

if (\_sender is MenuItem)

{

UpdateLentaProducts();

UpdateCardsCount();

}

return $"Добавить ({quant})";

}

}

else

MessageBox.Show(this, "Необходимо авторизоватся", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "Добавить";

});

var ACT\_DEL = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

if (Program.IsAuth())

{

if (MessageBox.Show(this, "Вы уверены что хотите убрать данный товар из корзины?", "Подтвердите", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

var del\_com = Program.SQL.CreateCommand();

del\_com.CommandText = $"DELETE FROM [ShoppingCart] WHERE [ProductID] = \'{IDP}\' AND [UserID] = \'{Program.GetUID()}\'";

int r = del\_com.ExecuteNonQuery();

UpdateLentaProducts();

UpdateCardsCount();

if(r == 0)

{

MessageBox.Show(this, "0 элементов затронуто", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

else

{

MessageBox.Show(this, "Данная позиция больше не в корзине", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

else

MessageBox.Show(this, "Необходимо авторизоватся", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return string.Empty;

});

var ACT\_EDIT = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

return string.Empty;

});

ViewerT.ViewerTable ctrl = new ViewerT.ViewerTable(prods, ACT\_ADD, ACT\_ABOUT, ACT\_EDIT, ACT\_DEL);

hostElementTable.Child = ctrl;

if (!Program.IsAuth())

{

foreach(var v in (ctrl.DataContext as ViewerT.ViewerTable\_ViewModel).MeProducts)

{

v.DelVisible = System.Windows.Visibility.Hidden;

v.EditVisible = System.Windows.Visibility.Hidden;

}

Debugger.Log(0, PATHCODE.ERROR, "SET HIDDEN");

}

}

/// <summary>

/// Событие для кнопки "Мой аккаунт"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void DeAuthToolStripMenuItem\_Click(object sender,EventArgs e)

{

Debugger.Log(505, PATHCODE.WARNING, "OPEN ME ACCOUNT", typeof(MainForm));

}

public void UpdateCardsCount()

{

if (Program.IsAuth())

{

int UID = Program.GetUID();

//ShoppingCart

var com = Program.SQL.CreateCommand();

com.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [UserID] = \'{UID}\'";

Debugger.Log(0, PATHCODE.MESSAGE, com.CommandText, typeof(MainForm));

using(var reader = com.ExecuteReader())

{

var tb = new DataTable();

tb.Load(reader);

cardsToolStripMenuItem.Text = $"Корзина ({tb.Rows.Count})";

}

}

}

private void EditToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void ExitAccountToolStripMenuItem\_Click(object sender, EventArgs e)

{

Properties.Settings.Default.Login = "";

Properties.Settings.Default.Password = "";

Properties.Settings.Default.SaveMe = false;

Properties.Settings.Default.Save();

}

private void cardsToolStripMenuItem\_Click(object sender, EventArgs e)

{

Dialogs.MeCardsForm mecard = new MeCardsForm();

mecard.Owner = this;

mecard.Show();

}

}

}

Код формы «Отзывы»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Ramen\_period.Dialogs

{

public partial class ProductReviewsForm : Form

{

public ProductReviewsForm(int IdProduct)

{

InitializeComponent();

BindL<ProductReviewsForm>.Set(this);

this.IdPRod = IdProduct;

}

private int IdPRod;

private void ProductReviewsForm\_Load(object sender, EventArgs e)

{

//Product Pre-load

var prod\_com = Program.SQL.CreateCommand();

prod\_com.CommandText = $"SELECT \* FROM [Products] WHERE [ProductID] = \'{IdPRod}\'";

var prod\_reader = prod\_com.ExecuteReader();

var tabProd = new DataTable();

tabProd.Load(prod\_reader);

prod\_reader.Close();

txtName.Text = tabProd.Rows[0].Field<string>("ProductName");

picProduct.Image = Program.GetBitmap(tabProd.Rows[0].Field<string>("ImageURL"));

txtDesc.Text = tabProd.Rows[0].Field<string>("Description");

var com = Program.SQL.CreateCommand();

com.CommandText = $"SELECT \* FROM [ProductReviews] WHERE [ProductID] = \'{IdPRod}\'";

var reader = com.ExecuteReader();

var tab = new DataTable();

tab.Load(reader);

reader.Close();

if(tab.Rows.Count > 0)

{

List<ViewerT.Reviews> ilist = new List<ViewerT.Reviews>();

for(int i = 0; i < tab.Rows.Count; i++)

{

//User Pre-load

var lite\_com = Program.SQL.CreateCommand();

string user\_id = tab.Rows[i].Field<int>("UserID").ToString();

lite\_com.CommandText = $"SELECT \* FROM [Users] WHERE [UserID] = \'{user\_id}\'";

var lite\_reader = lite\_com.ExecuteReader();

var lite\_tab = new DataTable();

lite\_tab.Load(lite\_reader);

lite\_reader.Close();

string user\_name = lite\_tab.Rows[0].Field<string>("FirstName");

var child = new ViewerT.Reviews()

{

FirstName = user\_name,

Comment = tab.Rows[i].Field<string>("Comment"),

Rating = tab.Rows[i].Field<int>("Rating").ToString(),

ReviewDate = tab.Rows[i].Field<DateTime>("ReviewDate")

};

Bitmap img\_rat = Properties.Resources.picStar\_Empty;

if(child.Rating == "1")

{

img\_rat = Properties.Resources.picStar\_1;

}

if (child.Rating == "2")

{

img\_rat = Properties.Resources.picStar\_2;

}

if (child.Rating == "3")

{

img\_rat = Properties.Resources.picStar\_3;

}

if (child.Rating == "4")

{

img\_rat = Properties.Resources.picStar\_4;

}

if (child.Rating == "5")

{

img\_rat = Properties.Resources.picStar\_5;

}

child.SetPicStar(img\_rat);

ilist.Add(child);

}

this.elemHost.Child = new ViewerT.ProductReviewsControl(ilist);

((this.elemHost.Child as ViewerT.ProductReviewsControl).DataContext as ViewerT.ProductReviews\_View).SetUrlImage(tabProd.Rows[0].Field<string>("ImageURL"));

}

else

{

MessageBox.Show(this, "Товар не имеет отзывы\nНапишите первыми!", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Abort;

}

private void btnSendRating\_Click(object sender, EventArgs e)

{

//TODO: HERE WE CREATE FORM FOR CREATE RATING

if (!Program.IsAuth())

{

MessageBox.Show(this,"Отзывы может оставлять только авторизованный пользователь!","Внимание!",MessageBoxButtons.OK,MessageBoxIcon.Information);

return;

}

}

}

}

Код формы «Корзина»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Ramen\_period.Dialogs

{

public partial class MeCardsForm : Form

{

public MeCardsForm()

{

InitializeComponent();

BindL<MeCardsForm>.Set(this);

}

private void MeCardsForm\_Load(object sender, EventArgs e)

{

var com = Program.SQL.CreateCommand();

com.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [UserID] = \'{Properties.Settings.Default.UserID}\'";

var reader = com.ExecuteReader();

var tb = new DataTable();

tb.Load(reader);

reader.Close();

decimal mepric = 0;

List<ViewerT.Card> ilist = new List<ViewerT.Card>();

for (int i = 0; i < tb.Rows.Count; i++)

{

ViewerT.Card card = new ViewerT.Card();

card.IdProduct = tb.Rows[i].Field<int>("ProductID");

var com\_prod = Program.SQL.CreateCommand();

com\_prod.CommandText = $"SELECT \* FROM [Products] WHERE [ProductID] = \'{card.IdProduct}\'";

Debugger.Log(0, PATHCODE.ERROR, $"{tb.Rows.Count}");

var reader\_prod = com\_prod.ExecuteReader();

var tb\_prod = new DataTable();

tb\_prod.Load(reader\_prod);

reader\_prod.Close();

var element = tb\_prod.Rows[0];

card.ProductName = element.Field<string>("ProductName");

card.Set\_Image\_Url(element.Field<string>("ImageURL"));

card.Description = element.Field<string>("Description");

card.AddedDate = tb.Rows[i].Field<DateTime>("AddedDate");

card.Quantity = tb.Rows[i].Field<int>("Quantity").ToString();

card.Price = element.Field<decimal>("Price").ToString();

mepric += element.Field<decimal>("Price") \* tb.Rows[i].Field<int>("Quantity");

ilist.Add(card);

}

ViewerT.HandlerProduct ACT\_ADD = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

var com\_check\_del = Program.SQL.CreateCommand();

com\_check\_del.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [UserID] = \'{Properties.Settings.Default.UserID}\' and [ProductID] = \'{IDP}\'";

var rd\_check\_del = com\_check\_del.ExecuteReader();

var tb\_check\_del = new DataTable();

tb\_check\_del.Load(rd\_check\_del);

rd\_check\_del.Close();

int QUANTITY = tb\_check\_del.Rows[0].Field<int>("Quantity");

QUANTITY++;

var com\_update\_del = Program.SQL.CreateCommand();

com\_update\_del.CommandText = $"UPDATE [ShoppingCart] Set [Quantity] = {QUANTITY} WHERE [UserID] = '{Properties.Settings.Default.UserID}' and [ProductID] = '{IDP}'";

int res = com\_update\_del.ExecuteNonQuery();

if (res == 0)

{

MessageBox.Show(this, "Ноль строк затронуто", "Рамен и точка", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

MeCardsForm\_Load(sender, e);

return QUANTITY.ToString();

});

ViewerT.HandlerProduct ACT\_DEL = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

var com\_check\_del = Program.SQL.CreateCommand();

com\_check\_del.CommandText = $"SELECT \* FROM [ShoppingCart] WHERE [UserID] = \'{Properties.Settings.Default.UserID}\' and [ProductID] = \'{IDP}\'";

var rd\_check\_del = com\_check\_del.ExecuteReader();

var tb\_check\_del = new DataTable();

tb\_check\_del.Load(rd\_check\_del);

rd\_check\_del.Close();

int QUANTITY = tb\_check\_del.Rows[0].Field<int>("Quantity");

QUANTITY--;

if (QUANTITY == 0)

{

var com\_update\_del = Program.SQL.CreateCommand();

com\_update\_del.CommandText = $"DELETE FROM [ShoppingCart] WHERE [UserID] = '{Properties.Settings.Default.UserID}' and [ProductID] = '{IDP}'";

int res = com\_update\_del.ExecuteNonQuery();

if (res == 0)

{

MessageBox.Show(this, "Ноль строк затронуто", "Рамен и точка", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

MeCardsForm\_Load(sender, e);

}

else

{

var com\_update\_del = Program.SQL.CreateCommand();

com\_update\_del.CommandText = $"UPDATE [ShoppingCart] Set [Quantity] = {QUANTITY} WHERE [UserID] = '{Properties.Settings.Default.UserID}' and [ProductID] = '{IDP}'";

int res = com\_update\_del.ExecuteNonQuery();

if (res == 0)

{

MessageBox.Show(this, "Ноль строк затронуто", "Рамен и точка", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

MeCardsForm\_Load(sender, e);

return QUANTITY.ToString();

}

return "0";

});

ViewerT.HandlerProduct ACT\_ORDER\_UPDATE = new ViewerT.HandlerProduct((IDP, \_sender) =>

{

return string.Empty;

});

elementHost1.Child = new ViewerT.CardsViewerControl(ilist, ACT\_DEL, ACT\_ADD, ACT\_ORDER\_UPDATE);

txt\_price.Text = $"Итог: {mepric} руб.";

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Abort;

}

}

}

Код формы «Авторизация»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Ramen\_period.Dialogs

{

public partial class AuthDialogForm : Form

{

/// <summary>

/// DialogResult start None

/// </summary>

public AuthDialogForm()

{

InitializeComponent();

txtlogin.Tag = "BAD";

BindL<AuthDialogForm>.Set(this);

}

private void btnBack\_Click(object sender, EventArgs e)

{

if(DialogResult == DialogResult.None)

{

this.DialogResult = DialogResult.Abort;

}

}

private void btnInvokeAuth\_Click(object sender, EventArgs e)

{

if(txtlogin.Tag.ToString() == "G")

{

if (!string.IsNullOrWhiteSpace(txtPass.Text))

{

var com = Program.SQL.CreateCommand();

com.CommandText = $"SELECT \* From [Users] WHERE [Phone] = \'+{txtlogin.Text}\' AND [Password] = \'{txtPass.Text}\'";

var reader = com.ExecuteReader();

DataTable tb = new DataTable();

tb.Load(reader);

if (tb.Rows.Count > 0)

{

reader.Close();

MessageBox.Show(this, "Авторизация успешна!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

Properties.Settings.Default.Login = txtlogin.Text;

Properties.Settings.Default.Password = txtPass.Text;

reader.Close();

Debugger.Log(0, PATHCODE.WARNING, $"{tb.Rows.Count}");

Properties.Settings.Default.UserID = tb.Rows.Cast<DataRow>().FirstOrDefault().Field<int>("UserID");

if (cb\_GetMe.Checked)

Properties.Settings.Default.Save();

DialogResult = DialogResult.Abort;

}

else

{

reader.Close();

MessageBox.Show(this, "Не удалось войти", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

else

{

MessageBox.Show(this, "Ошибка данных!\nВ поле логин", "Введите корректировки", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void txtlogin\_TextChanged(object sender, EventArgs e)

{

double tmp;

if (txtlogin.Text.Length <= 0)

{

txtlogin.BackColor = Color.FromArgb(255, 255, 255);

txtlogin.Tag = "BAD";

return;

}

else if ((txtlogin.Text.Length < "79001234561".Length || txtlogin.Text.Length > "79001234561".Length) || !double.TryParse(txtlogin.Text,out tmp))

{

txtlogin.BackColor = Color.FromArgb(255, 128, 128);

txtlogin.Tag = "BAD";

}

else

{

txtlogin.BackColor = Color.FromArgb(255, 255,255);

txtlogin.Tag = "G";

}

}

private void AuthDialogForm\_Load(object sender, EventArgs e)

{

txtPass.UseSystemPasswordChar = !cb\_showpass.Checked;

cb\_GetMe.Checked = Properties.Settings.Default.SaveMe;

if (cb\_GetMe.Checked)

{

txtlogin.Text = Properties.Settings.Default.Login;

txtPass.Text = Properties.Settings.Default.Password;

}

}

private void cb\_showpass\_CheckedChanged(object sender, EventArgs e)

{

txtPass.UseSystemPasswordChar = !cb\_showpass.Checked;

}

private void AuthDialogForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Properties.Settings.Default.SaveMe = cb\_GetMe.Checked;

Properties.Settings.Default.Save();

}

}

}

Код формы «О продукте»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Ramen\_period.Dialogs

{

public partial class AboutProductForm : Form

{

public AboutProductForm(int index)

{

InitializeComponent();

IndexProduct = index;

BindL<AboutProductForm>.Set(this);

}

private int IndexProduct;

private void AboutProductForm\_Load(object sender, EventArgs e)

{

var com = Program.SQL.CreateCommand();

com.CommandText = $"SELECT \* FROM [Products] WHERE [ProductID] = \'{IndexProduct}\'";

var reader = com.ExecuteReader();

var tb = new DataTable();

tb.Load(reader);

reader.Close();

var element = tb.Rows[0];

if(element != null)

{

txtDescr.Text = element.Field<string>("Description");

txt\_name.Text = element.Field<string>("ProductName");

txt\_addeddate.Text = element.Field<DateTime>("AddedDate").ToString();

img\_prod.Image = Program.GetBitmap(element.Field<string>("ImageURL"));

txt\_price.Text = element.Field<Decimal>("Price").ToString() + " руб.";

var com\_cat = Program.SQL.CreateCommand();

com\_cat.CommandText = $"SELECT \* FROM [Categories] WHERE [CategoryID] = \'{element.Field<int>("CategoryID")}\'";

var reader\_cat = com\_cat.ExecuteReader();

var tb\_cat = new DataTable();

tb\_cat.Load(reader\_cat);

reader\_cat.Close();

txt\_categories.Text = tb\_cat.Rows[0].Field<string>("CategoryName");

}

else

{

Debugger.Log(0, PATHCODE.ERROR, "element was null!", typeof(AboutProductForm));

}

}

private void btnBack\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.Abort;

}

}

}

Код главного элемента окна

<UserControl x:Class="ViewerT.ViewerTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:ViewerT"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<ListBox SelectionMode="Single" Name="list" ItemsSource="{Binding MeProducts}" Background="#6E8898" SelectedItem="{Binding SelectProduct}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Setter Property="IsSelected" Value="{Binding Content.IsSelected, Mode=TwoWay, RelativeSource={RelativeSource Self}}"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListBoxItem">

<ContentPresenter/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</ListBox.ItemContainerStyle>

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Width="800" Orientation="Vertical" HorizontalAlignment="Center" MouseEnter="StackPanel\_MouseEnter">

<TextBlock Margin="0 20 0 0" HorizontalAlignment="Center" Text="{Binding ProductName}"/>

<Image Margin="0 5 0 0" HorizontalAlignment="Center" Width="160" Height="160" Source="{Binding ImageProduct}"/>

<StackPanel Margin="0 5 0 0" Orientation="Horizontal" HorizontalAlignment="Center">

<Button Name="btn\_about" Tag="{Binding IdProduct}" Click="btn\_about\_Click" Content="О продукте" HorizontalAlignment="Center" Width="70" BorderThickness="1" BorderBrush="#E27D60" Background="#F8F0E3">

</Button>

<Button Name="btn\_add" Tag="{Binding IdProduct}" Click="btn\_add\_Click" Margin="5 0 0 0" Content="{Binding CountAdded}" HorizontalAlignment="Center" Width="90" BorderThickness="1" BorderBrush="#E27D60" Background="#F8F0E3">

</Button>

<StackPanel>

<Button Name="btn\_del" Tag="{Binding IdProduct}" Click="btn\_del\_Click" Margin="5 0 0 0" Content="Удалить" Visibility="{Binding DelVisible,Mode=OneWay}" HorizontalAlignment="Center" Width="90" BorderThickness="1" BorderBrush="#E27D60" Background="#F8F0E3">

</Button>

<Button Name="btn\_edit" Tag="{Binding IdProduct}" Click="btn\_edit\_Click" Margin="5 0 0 0" Content="Изменить" Visibility="{Binding EditVisible,Mode=OneWay}" HorizontalAlignment="Center" Width="90" BorderThickness="1" BorderBrush="#E27D60" Background="#F8F0E3">

</Button>

</StackPanel>

</StackPanel>

<TextBlock Margin="0 5 0 0" HorizontalAlignment="Center" Text="{Binding Description}"/>

<TextBlock Margin="0 5 0 0" HorizontalAlignment="Center" Text="{Binding Price}"/>

<TextBlock Margin="0 5 0 0" HorizontalAlignment="Center" Text="{Binding AddedDate}"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.ContextMenu>

<ContextMenu >

<MenuItem Header="Отзывы" Click="btn\_about\_Click"/>

<MenuItem Header="Добавить в корзину" Click="btn\_add\_Click"/>

</ContextMenu>

</ListBox.ContextMenu>

</ListBox>

</Grid>

</UserControl>

C#

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Drawing.Imaging;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Automation.Peers;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ViewerT

{

/// <summary>

/// Логика взаимодействия для ViewerTable.xaml

/// </summary>

public partial class ViewerTable : UserControl

{

private HandlerProduct ACT\_ADD;

private HandlerProduct ACT\_DEL;

private HandlerProduct ACT\_EDIT;

private HandlerProduct ACT\_ABOUT;

public ViewerTable(List<MeProducts> products, HandlerProduct act\_add,HandlerProduct act\_about, HandlerProduct act\_edit, HandlerProduct act\_del)

{

InitializeComponent();

this.ACT\_ADD = act\_add;

this.ACT\_DEL = act\_del;

this.ACT\_ABOUT = act\_about;

this.ACT\_EDIT = act\_edit;

this.DataContext = new ViewerTable\_ViewModel(products);

}

private void btn\_add\_Click(object sender, RoutedEventArgs e)

{

if(sender is Button)

{

string update\_context = ACT\_ADD?.Invoke(int.Parse((sender as Button).Tag.ToString()),sender);

if (update\_context != string.Empty)

{

(sender as Button).Content = update\_context;

}

}

else if(sender is MenuItem)

{

ACT\_ADD?.Invoke(MouseProduct.IdProduct,sender);

}

}

private MeProducts MouseProduct;

private void btn\_about\_Click(object sender, RoutedEventArgs e)

{

if (sender is Button)

{

ACT\_ABOUT?.Invoke(int.Parse((sender as Button).Tag.ToString()), sender);

}

else if (sender is MenuItem)

{

ACT\_ABOUT?.Invoke(MouseProduct.IdProduct, sender);

}

}

private void StackPanel\_MouseEnter(object sender, MouseEventArgs e)

{

MouseProduct = (sender as StackPanel).DataContext as MeProducts;

}

private void btn\_del\_Click(object sender, RoutedEventArgs e)

{

if(sender is Button)

{

ACT\_DEL?.Invoke(int.Parse((sender as Button).Tag.ToString()), sender);

}

else if (sender is MenuItem)

{

ACT\_DEL?.Invoke(MouseProduct.IdProduct, sender);

}

}

private void btn\_edit\_Click(object sender, RoutedEventArgs e)

{

}

}

public delegate string HandlerProduct(int id\_product,object sender);

public class MeProducts : INotifyPropertyChanged

{

/// <summary>

/// Метод получение изображения

/// </summary>

/// <param name="IURL">URL - Адрес изображение как локальный так и из сети</param>

/// <returns></returns>

private static Bitmap GetBitmap(string IURL)

{

try

{

//Экземпляр WebClient (Встроенный класс для примитивной работы с HTTP протоколом)

WebClient web = new WebClient();

//Скачиваем массив байтов по ссылке (И не важно удалённый или локальный адрес)

byte[] dat = web.DownloadData(IURL);

//Массив байт переводим в поток MemoryStream из System.IO и используем метод FromStream класса Image

//И делаем привидение типа Image к Bitmap (полеморфизм) и возращаем Bitmap

return (Bitmap)System.Drawing.Image.FromStream(new MemoryStream(dat));

}

//Исключение если мало прав (нету доступа для выделения потока)

catch (NotSupportedException g)

{

// Debugger.Log(1, PATHCODE.WARNING, $"Ошибка потока. Войдите как администратор.\n{g}");

}

//Исключение нету интернета или доступа фаервол

catch (WebException g)

{

// Debugger.Log(1, PATHCODE.WARNING, $"Ошибка подключение к интернету!\n{g}");

}

//Остальные исключение которые могут возникнуть, не вызываем CRITICAL а вернём избр. вшитое

catch (Exception g)

{

// Debugger.Log(0, PATHCODE.ERROR, g, typeof(Program));

}

//Возвращаяем избр. по умолчанию

return null;

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged(string propertyName) => PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

private string image\_url;

public BitmapImage ImageProduct

{

get

{

BitmapImage bt = new BitmapImage();

var bitm = GetBitmap(image\_url);

using (MemoryStream memory = new MemoryStream())

{

bitm.Save(memory, ImageFormat.Png);

memory.Position = 0;

bt = new BitmapImage();

bt.BeginInit();

bt.StreamSource = memory;

bt.CacheOption = BitmapCacheOption.OnLoad;

bt.EndInit();

}

return bt;

}

set

{

OnPropertyChanged("ImageProduct");

}

}

private int \_IdProduct;

public int IdProduct

{

get

{

return \_IdProduct;

}

set

{

OnPropertyChanged("IdProduct");

\_IdProduct = value;

}

}

private int \_CountAdded;

private Visibility \_DelBtn = Visibility.Visible;

public Visibility DelVisible

{

get

{

return \_DelBtn;

}

set

{

OnPropertyChanged("DelVisible");

\_DelBtn = value;

}

}

private Visibility \_EditVisible = Visibility.Visible;

public Visibility EditVisible

{

get

{

return \_EditVisible;

}

set

{

OnPropertyChanged("EditVisible");

\_EditVisible = value;

}

}

public string CountAdded

{

get

{

return $"Добавить ({\_CountAdded})";

}

set

{

OnPropertyChanged("CountAdded");

\_CountAdded = int.Parse(value);

}

}

public void Set\_Image\_Url(string url)

{

image\_url = url;

}

private string \_ProductName;

public string ProductName

{

get

{

return \_ProductName;

}

set

{

OnPropertyChanged("ProductName");

\_ProductName = value;

}

}

private string \_Description;

public string Description

{

get

{

return \_Description;

}

set

{

OnPropertyChanged("Description");

\_Description = value;

}

}

private string \_Price;

public string Price

{

get

{

return \_Price + " руб.";

}

set

{

OnPropertyChanged("Price");

\_Price = value;

}

}

private string \_AddedDate;

public DateTime AddedDate

{

get

{

return DateTime.Parse(\_AddedDate);

}

set

{

OnPropertyChanged("AddedDate");

\_AddedDate = value.ToShortDateString();

}

}

}

public class ViewerTable\_ViewModel : INotifyPropertyChanged

{

public ObservableCollection<MeProducts> MeProducts { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

private MeProducts \_SelectProduct;

public MeProducts SelectProduct

{

get

{

return \_SelectProduct;

}

set

{

OnPropertyChanged("SelectProduct");

\_SelectProduct = value;

}

}

public ViewerTable\_ViewModel(List<MeProducts> meProducts)

{

MeProducts = new ObservableCollection<MeProducts>();

MeProducts.Clear();

foreach(var v in meProducts)

{

MeProducts.Add(v);

}

}

public void OnPropertyChanged(string propertyName) => PropertyChanged?.Invoke(propertyName, new PropertyChangedEventArgs(propertyName));

}

}

**ЗАКЛЮЧЕНИЕ**

Основная информация

Название приложения: Рамен и точка

Цель приложения: Помочь работникам ресторана предоставить учёт, хранения, обработку заказов, информации о продуктов и т.д.

Функции: Для пользователей есть возможность просматривать записи. Для администраторов кроме возможности просмотра записей, так есть могут редактировать информацию (добавлять, удалять записи).

Результаты

Разработка приложения была выполнена успешно;

Цели и задачи были достигнуты и выполнены;

Разработка базы данных и приложения для Рамен и точка - это был интересный и познавательный опыт. Я узнал много нового о структурировании данных, оптимизации запросов и разработке удобного пользовательского интерфейса.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Jacob VanWagoner, Modern SQL: A Practical Guide for Working with Relational Databases (2021, 400 стр.)
2. Джозеф Л. Форджоне, SQL Server 2022: A Beginner's Guide (2022, 592 стр.)
3. Джонатан Кларк, C# in Depth (2021, 1008 стр.)
4. Дэвид Хэмилтон, Visual Studio 2022: A Beginner's Guide (2022, 576 стр.)
5. Иэн Гриффитс, Mastering Visual Studio 2022 (2022, 960 стр.)
6. Крис Феа, SQL for Data Analysis (2020, 432 стр.)
7. Кэтти Сиэрс и Билл МакКоун, Head First C# (2018, 752 стр.)
8. Линн Бейли, Билл МакКоун и Кэтти Сиэрс, Head First SQL (2018, 768 стр.)
9. Роберт МакГрат и Боб Тавелла, Professional SQL Server 2019 (2019, 1280 стр.)
10. Роберт Уайлднер, Pro Visual Studio 2019: The Expert's Guide to Microsoft's Integrated Development Environment (2019, 1344 стр.)
11. Мартин Фаулер, Refactoring: Improving the Design of Existing Code (2018, 464 стр.)
12. Стивен К. МакКоннелл, Code Complete: A Practical Handbook of Software Construction (2004, 960 стр.)
13. Брэдфорд Т. Форд, SQL Cookbook (2020, 864 стр.)
14. Бен Гриффитс и Дэн Рат, Programming Microsoft SQL Server 2019 (2019, 1056 стр.)
15. Скотт Хансельман, The ASP.NET Core 6.0 MVC Cheat Sheet (2021, 120 стр.)
16. Мик С. Доэрти и Шон Б. Шеннон, Applied Microsoft SQL Server 2019 Analysis Services (2019, 672 стр.)
17. Дон Джонс, Learn SQL Quickly (2019, 160 стр.)