ДЕПАРТАМЕНТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ТОМСКОЙ ОБЛАСТИ
ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.07 Информационные системы и программирование

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «КОФЕ ПЬЁМ»

Пояснительная записка
к курсовому проекту
КП.23.09.02.07.602.08.ПЗ

Студент
«__»_____ 2023 г. _____ В. А. Кипоров
Руководитель
«__»_____ 2023 г. _____ А. Ю. Маюнова

Томск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире информационные системы становятся неотъемлемой частью различных сфер деятельности, включая сферу общественного питания. Кофейни, столовые, рестораны и кафе являются популярными местами, где люди проводят время и наслаждаются атмосферой, ароматом свежего кофе и вкусными закусками. Однако, с ростом популярности заведений общественного питания, владельцы сталкиваются с проблемами, связанными с управлением заказами, обслуживанием клиентов и эффективным учетом финансовых операций.

В рамках данного курсового проекта была разработана информационная система для кофейни, с фокусом на создание кассы самообслуживания. Касса самообслуживания представляет собой решение, которое позволяет клиентам самостоятельно оформлять заказы и производить оплату без необходимости обращаться к персоналу заведения. Это удобно и экономит время как для посетителей, так и для сотрудников заведения.

Целью данного проекта является улучшение процесса обслуживания клиентов в кофейне, повышение эффективности работы персонала и увеличение общей прибыльности заведения путём ускорения обслуживания клиентов и исключением официантского состава. Информационная система для кофейни включает в себя не только кассу самообслуживания, но и модули для управления меню, сотрудниками, акциями и учетом финансовых операций.

Ожидается, что разработанная информационная система для кофейни с кассой самообслуживания будет способствовать повышению уровня обслуживания клиентов и оптимизации работы персонала.

1 ОБЩАЯ ЧАСТЬ

1.1 Анализ предметной области

1.1.1 Описание функционала

В рамках данного проекта необходимо разработать информационную систему для упрощения бизнес-процессов кофейни. Система должна предоставлять пользователю следующие возможности:

- Выбор блюд по категориям
- Изменение количества блюд
- Добавление блюда в корзину
- Удаление блюда из корзины
- Очистка корзины
- Создание заказа
- Просмотр чека покупки
- Просмотр готовности заказа
- Просмотр акций
- Добавление блюд, пользователей, акций и категорий
- Удаление блюд, пользователей, акций и категорий
- Просмотр блюд, пользователей, акций и категорий
- Создание отчётов в excel-таблицу
- Просмотр профиля и смена пароля для авторизованных пользователей
- Авторизация
- Деаутентификация
- Просмотр заказов в соответствие со статусами
- Просмотр готовых заказов за текущую смену
- Изменение статуса заказа

- Просмотр блюд в заказе

А также система должна выполнять автоматически:

- Расчёт цены блюда с учётом акции
- Изменение статуса заказа по таймеру
- Изменение статуса акции в зависимости от текущей даты
- При выборе изображения оно добавляется в систему
- Сортировка заказов по месяцам при создании excel-отчёта

База данных информационной системы должна хранить в себе следующие данные:

- Пользователей (логин, пароль, имя, фамилия, отчество и должность);
- Блюда (название, цена, фото и категория);
- Категории (название и фото);
- Акции (блюдо, дата окончания, процент скидки, итоговая стоимость блюда);
- Заказы (список блюд, дата и время, итоговая цена и статус);
- Историю заказов (заказ, повар, дата и время приёма заказа в готовку).

1.1.2 Анализ конкурентов

1.1.2.1 Облачная информационная система «Presto»

Сайт: https://sbis.ru/presto

Плюсы:

- Возможность ведения бухгалтерии
- Кроссплатформенность
- Использование карт сотрудников

Минусы:

- Находится в стадии разработки

1.1.2.2 «Iiko»

Сайт: https://joinposter.com/

Плюсы:

- Простой интерфейс
- Отзывчивая поддержка
- Удобный мониторинг финансов
- Низкая цена в сравнении с конкурентами
- Стабильность

Минусы:

- Закрыт доступ к базе
- Не работает на территории России и Белоруссии
- Невозможность установки разных тарифов для разных заведений

1.1.2.3 Облачная информационная система «Poster»

Сайт: https://joinposter.com/

Плюсы:

- Простой интерфейс
- Отзывчивая поддержка
- Удобный мониторинг финансов
- Низкая цена в сравнении с конкурентами
- Стабильность

Минусы:

- Закрыт доступ к базе
- Не работает на территории России и Белоруссии
- Невозможность установки разных тарифов для разных заведений

1.2 Средства и среды разработки

Для разработки были выбраны следующие инструменты:

1.2.1 Draw.io – это бесплатный онлайн-сервис для создания диаграмм и схем любой сложности. С помощью данного сервиса можно легко создавать блок - схемы, организационные диаграммы, UML диаграммы, ER-диаграммы, схемы баз данных, а также многие другие типы диаграмм.

1.2.2 JetBrains Rider – это кросс-платформенная IDE для .NET-разработчиков, основанная на платформе IntelliJ и ReSharper, которая отличается высокой производительностью от своего ближайшего конкурента в лице Microsoft Visual Studio.

1.2.3 Выбирая между огромным множеством СУБД, я остановился на PostgreSQL, ввиду простоты использования при помощи фреймворков и очень удобной структуры баз данных.

1.2.4 При выборе языка программирования конкурировали 2 языка программирования: 1С и C#. Мой выбор пал на C# ввиду следующих причин:

- C# позволяет гибко настаивать интерфейс.
- На C# можно реализовать более оптимизированное приложение.
- Программу на C# можно наделить собственной лицензией.
- C# в отличие от 1С востребован во всём мире

1.2.5 В проекте использовалось всего два фреймворка: AvaloniaUI для реализации интерфейсов, и EntityFramework Core для удобной реализации базы данных в коде.

2 СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Описание требований к информационной системе

2.1.1 Список терминов и их определения

Пользователь – это лицо или организация, которое использует действующую систему для выполнения конкретной функции.

База данных – это упорядоченный набор структурированной информации или данных, которые хранятся в электронном виде в компьютерной системе.

Администратор – пользователь, который имеет расширенные права, такие как добавление, удаление, и просмотра записей.

Авторизация – это процесс проверки данных пользователя, при успешном завершении которого пользователь получает доступ к определённому перечню действий.

Деаутентификация – это процесс по завершение которого, пользователь перестаёт иметь доступ к определённому перечню действий, полученного в ходе авторизации.

Логирование - процесс записи в хронологическом порядке событий, происходящих с каким-то объектом или в рамках какого-то процесса, в специальный файл или базу данных.

2.1.2 Требования

Информационная система должна обеспечивать следующие возможности:

2.1.2.1 Для покупателя

- Выбор категории
- Указание количества блюда
- Добавление блюда в корзину
- Удаление блюда из корзины

- Очистка корзины
- Создание заказа
- Показ чека покупки
- Система отслеживания готовности заказа
- Показ акций

2.1.2.2 Для администратора

- Авторизация
- Добавление блюд, пользователей, акций и категорий
- Удаление блюд, пользователей, акций и категорий
- Просмотр заказов, блюд, пользователей, акций и категорий
- Создание отчётов в excel-таблицу
- Просмотр профиля и смена пароля
- Деаутентификация

2.1.2.3 Для повара

- Авторизация
- Просмотр заказов со статусом «Оплачено»
- Просмотр выполненных заказов за смену
- Изменение статуса заказа
- Просмотр блюд в заказе

2.1.2.4 На стороне системы

- Расчёт цены блюда с учётом акции
- Изменение статуса заказа по таймеру
- Изменение статуса акций в зависимости от текущей даты
- При выборе изображения оно добавляется в систему
- Сортировка заказов по месяцам при создании excel-отчёта

2.2 Диаграммы вариантов использования

В рамках данной информационной системы существует 3 пользователя: неавторизованный, повар и администратор. Каждый из них имеет широкий функционал и описан следующими диаграммами вариантов использования.

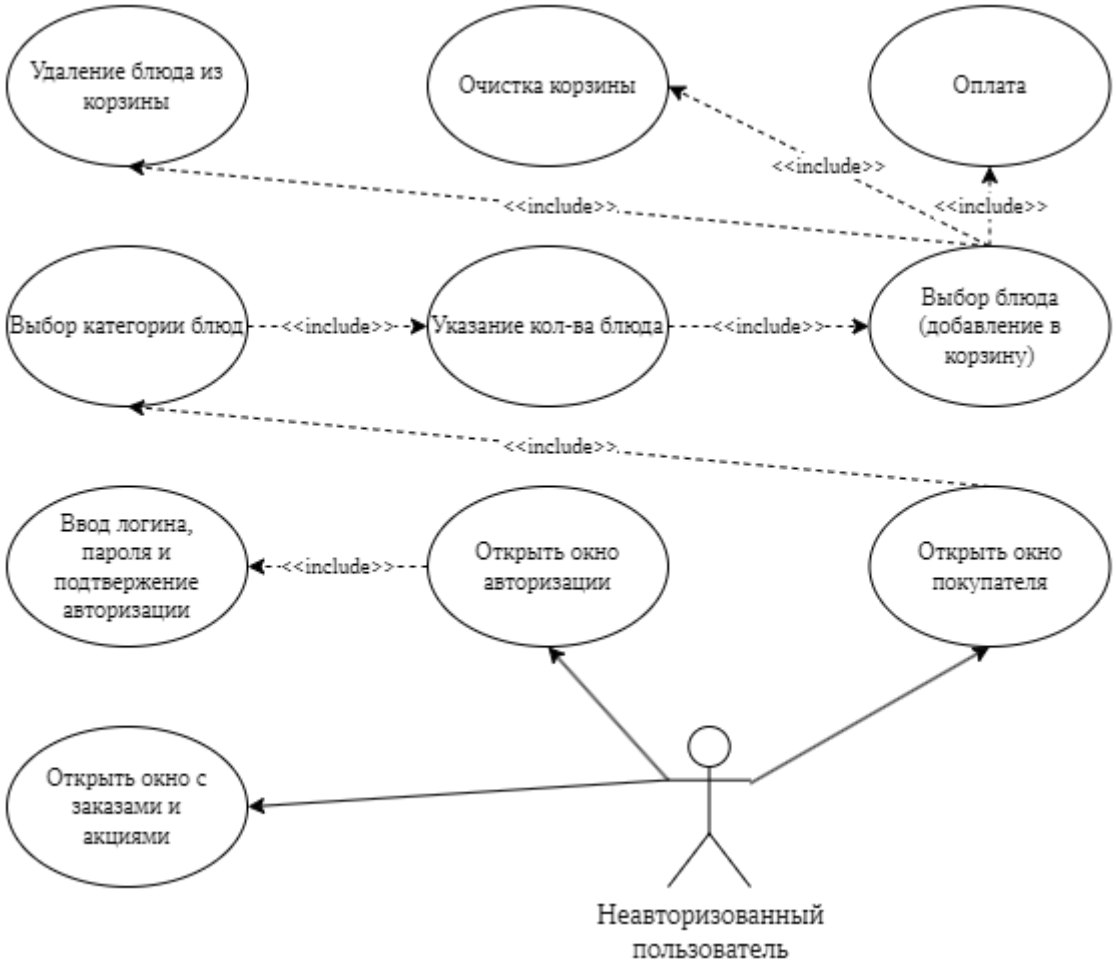


Рисунок 2.2.1 – Диаграмма вариантов использования (неавторизованный пользователь)

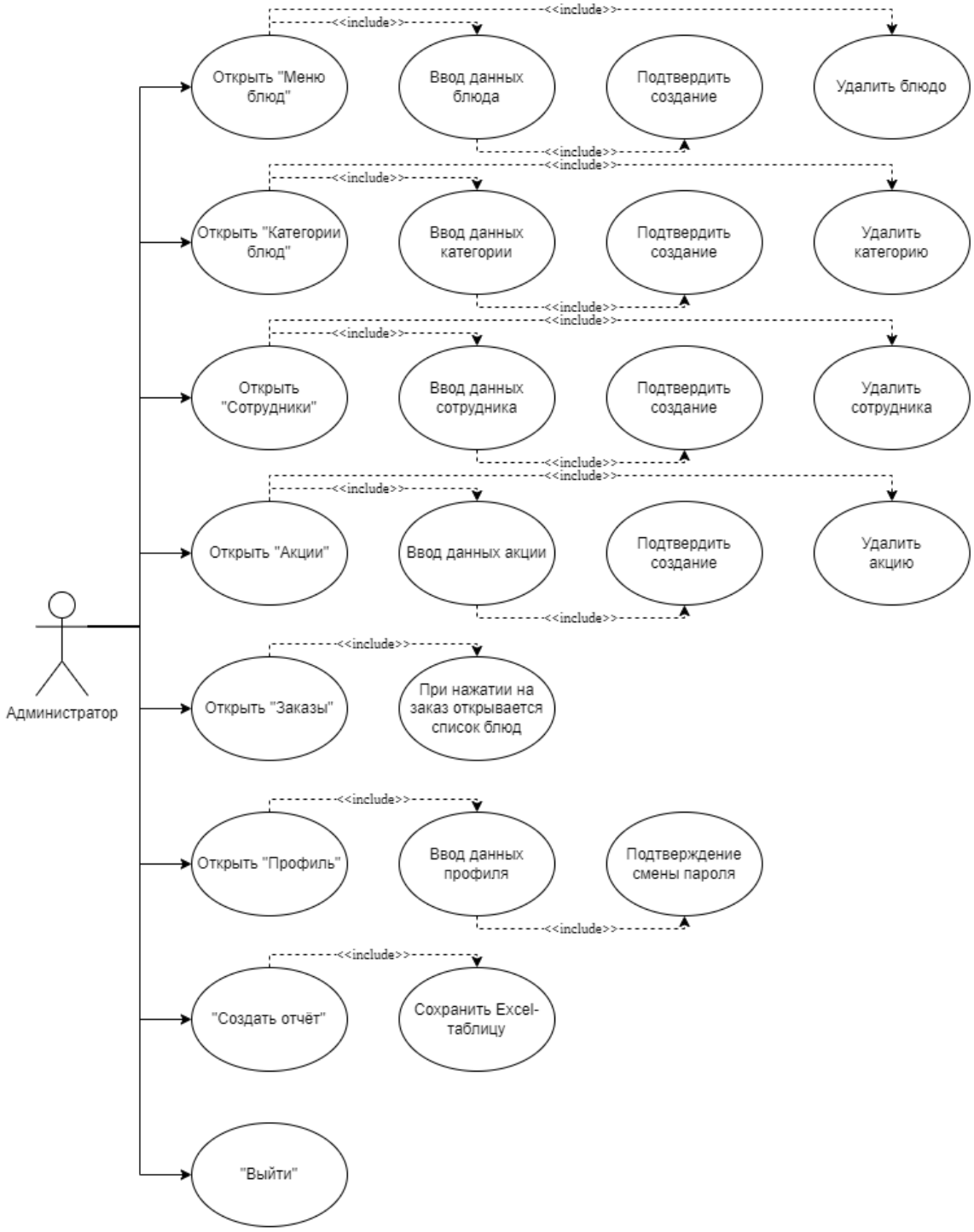


Рисунок 2.2.2 – Диаграмма вариантов использования (пользователь авторизован как администратор)

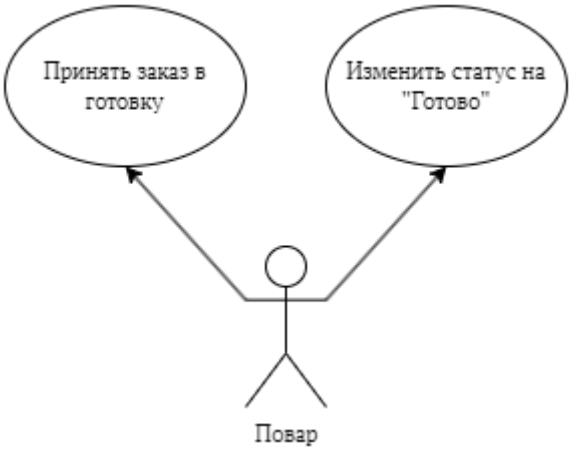


Рисунок 2.2.3 – Диаграмма вариантов использования (пользователь авторизован как повар)

2.3 Диаграммы состояний

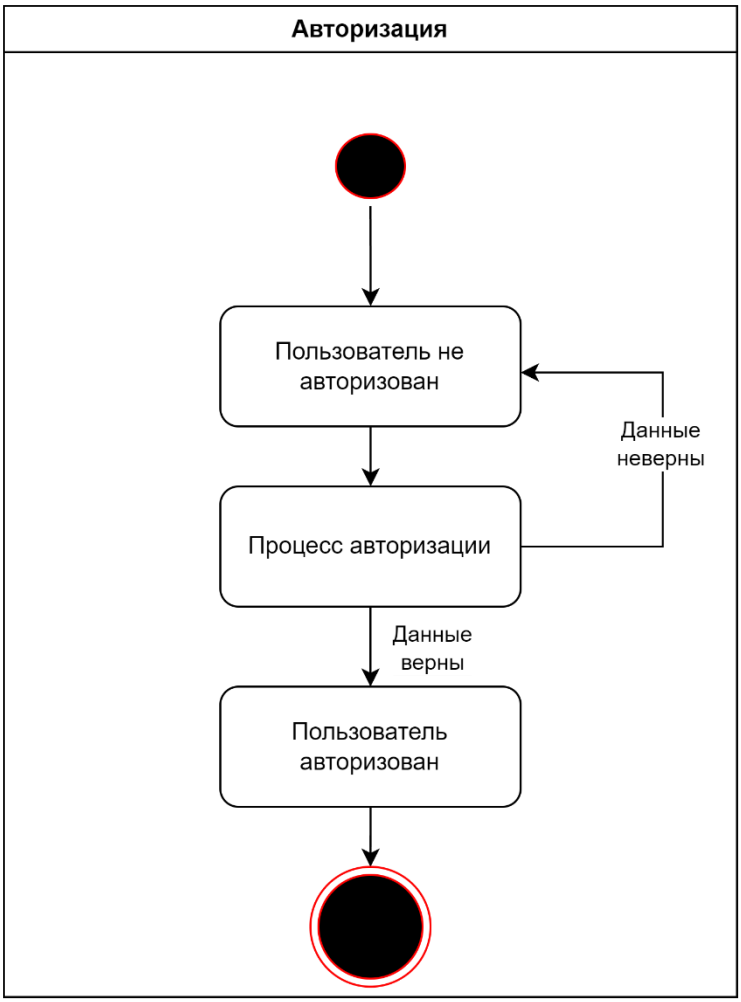


Рисунок 2.3.1 - Диаграмма состояний функции «Авторизация»

Рисунок 2.3.2 – Диаграмма состояний функции «Добавление блюда»

Рисунок 2.3.3 – Диаграмма состояний функции «Добавление пользователя»

Рисунок 2.3.4 – Диаграмма состояний функции «Добавление категории»

Рисунок 2.3.5 – Диаграмма состояний функции «Добавление акции»

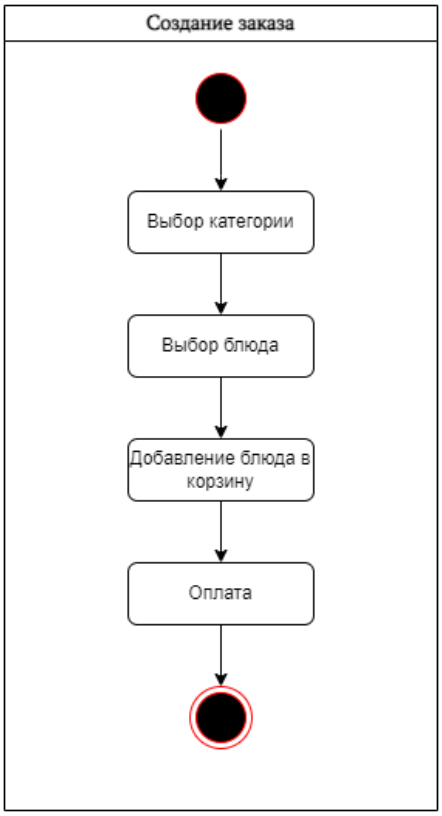


Рисунок 2.3.6 – Диаграмма состояний функции «Создание заказа»

Рисунок 2.3.7 – Диаграмма состояний функции «Удаление категории»

Рисунок 2.3.8 - Диаграмма состояний функции «Удаление акции»

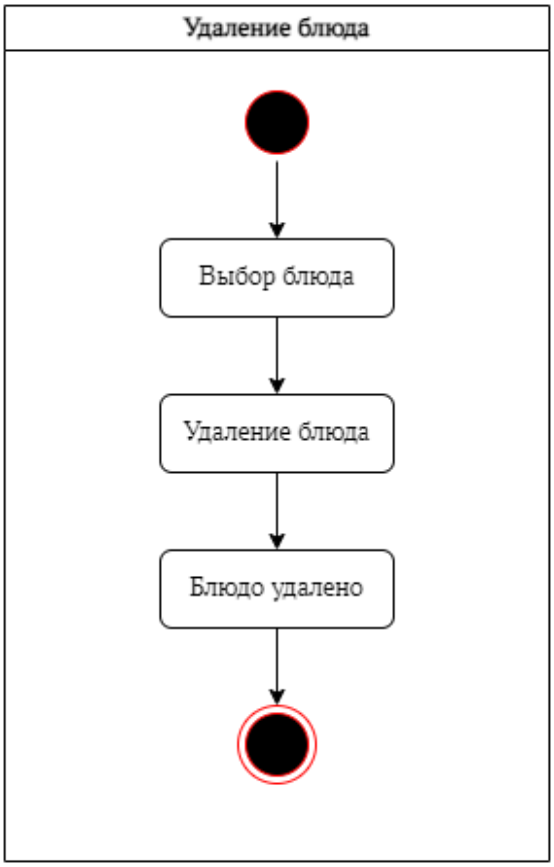


Рисунок 2.3.9 - Диаграмма состояний функции «Удаление блюда»

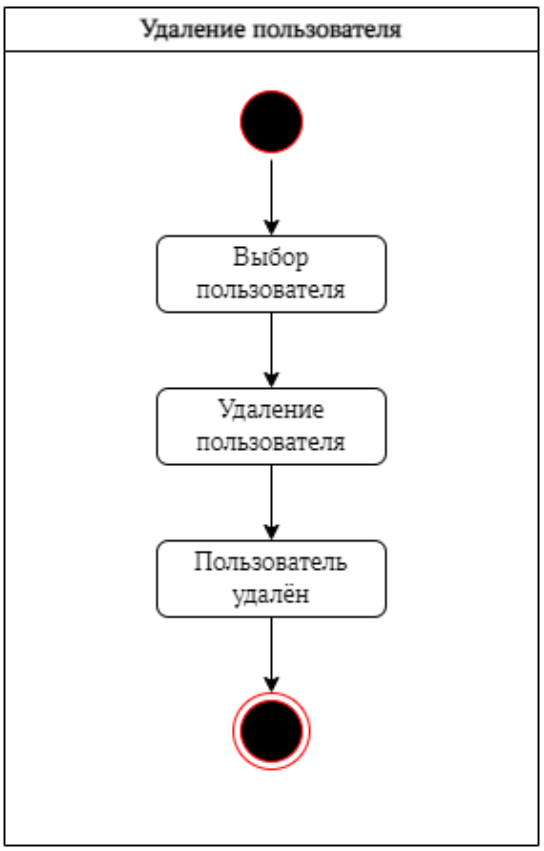


Рисунок 2.3.10 - Диаграмма состояний функции «Удаление пользователя»

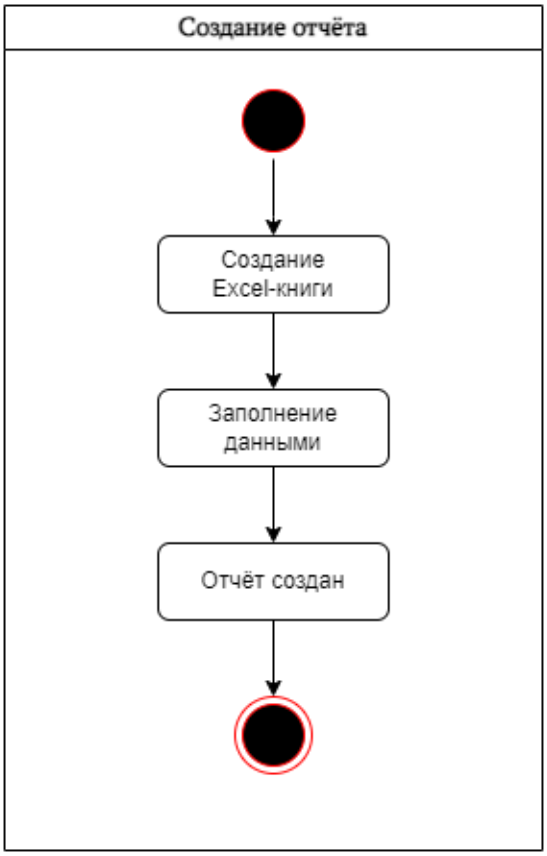


Рисунок 2.3.11 - Диаграмма состояний функции «Создание отчёта»

Рисунок 2.3.12 - Диаграмма состояний функции «Изменение статуса заказа»

Рисунок 2.3.13 - Диаграмма состояний функции «Изменение статуса заказа на «Выдано»»

2.4 Схема данных

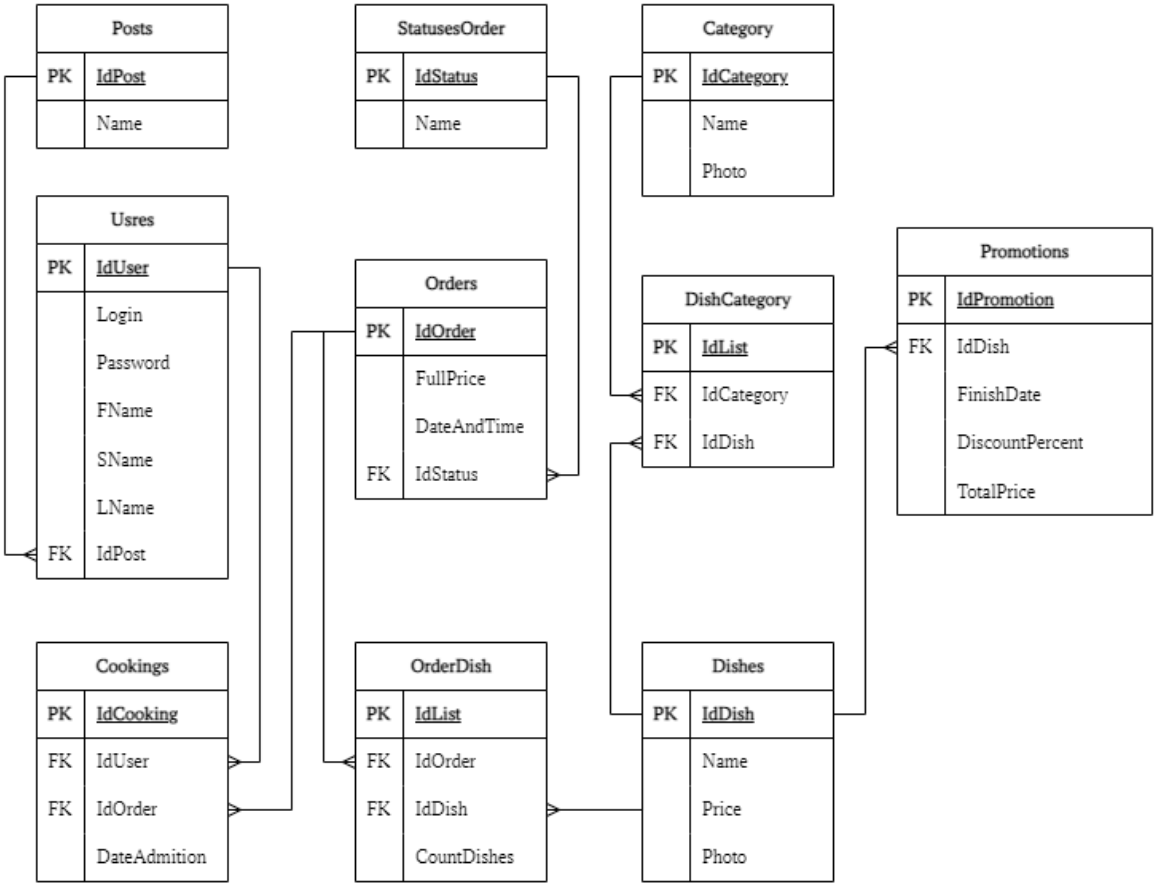


Рисунок 2.4.1 – Логическая модель базы данных

Таблица 2.4.1 – Users

№ п-п	Название поля	Тип данных	Описание
1	IdUser	int	Первичный ключ
2	Login	varchar(30)	Логин
3	Password	varchar(30)	Пароль
4	FName	varchar(30)	Имя
5	LName	varchar(30)	Отчество
6	MName	varchar(30)	Фамилия
7	IdPost	int	Ссылка на должность

Таблица 2.4.2 - Posts

№ п-п	Название поля	Тип данных	Описание
1	IdPost	int	Первичный ключ
2	Name	varchar(30)	Название должности

Таблица 2.4.3 – Cookings

№ п-п	Название поля	Тип данных	Описание
1	IdCooking	int	Первичный ключ
2	IdUser	int	Ссылка на пользователя
3	IdOrder	int	Ссылка на заказ
4	DateAdmission	timestamp	Дата приёма заказа в изготовление

Таблица 2.4.4 – Orders

№ п-п	Название поля	Тип данных	Описание
1	IdOrder	int	Первичный ключ
2	FullPrice	float	Цена заказа
3	DateAndTime	timestamp	Дата создания
4	IdStatus	int	Ссылка на статус

Таблица 2.4.5 – StatusesOrder

№ п-п	Название поля	Тип данных	Описание
1	IdStatus	int	Первичный ключ
2	Name	varchar(30)	Название статуса

Таблица 2.4.6 – OrderDish

№ п-п	Название поля	Тип данных	Описание
1	IdList	int	Первичный ключ
2	IdOrder	int	Ссылка на заказ
3	IdDish	int	Ссылка на блюдо
4	CountDishes	int	Количество одинаковых блюд в заказе

Таблица 2.4.7 – Dishes

№ п-п	Название поля	Тип данных	Описание
1	IdDish	int	Первичный ключ
2	Price	float	Цена
3	Photo	varchar(250)	Путь до изображения
4	Name	varchar(30)	Название блюда

Таблица 2.4.8 – DishCategory

№ п-п	Название поля	Тип данных	Описание
1	IdList	int	Первичный ключ
2	IdCategory	float	Ссылка на категорию
3	IdDish	timestamp	Ссылка на блюдо

Таблица 2.4.9 – Category

№ п-п	Название поля	Тип данных	Описание
1	IdCategory	int	Первичный ключ
2	Name	varchar(30)	Название
3	Photo	varchar(250)	Путь до изображения

Таблица 2.4.10 – Promotions

№ п-п	Название поля	Тип данных	Описание
1	IdPromotion	int	Первичный ключ
2	IdDish	int	Ссылка на блюдо
3	FinishDate	timestamp	Дата окончания акции
4	DiscountPercent	int	Процент скидки
5	TotalPrice	float	Итоговая цена блюда

2.5 Пользовательские сценарии

2.5.1 Пользовательский сценарий неавторизованного пользователя

Пользователь запускает информационную систему.

В главном окне он может открыть «Окно заказов и акций», «Окно авторизации» и «Окно покупателя».

В окне авторизации пользователь может пройти авторизацию, введя свои данные, или вернуться назад.

В окне покупателя пользователь может выбрать категорию блюд, добавить блюдо в корзину, изменить его количество, удалить блюдо из корзины, очистить корзину и оформить заказ.

2.5.2 Пользовательский сценарий для администратора

После авторизации в качестве администратора, пользователя встречает окно с множеством страниц, которые открываются по следующим, одноимённым кнопкам: «Меню блюд», «Категории блюд», «Сотрудники», «Акции», «Заказы», «Профиль». А также в данном окне присутствуют кнопки «Создать отчёт», которая создаёт Excel-таблицу с заказами за этот год, «Выйти», которая деаутентифицирует пользователя и переносит его в «Окно авторизации». Если пользователь откроет станицу «Меню блюд», то в нём он сможет посмотреть существующие блюда, удалить блюдо и добавить блюдо. Если пользователь откроет страницу «Категории блюд», то откроется одноимённая страница, в которой пользователь может посмотреть существующие категории, удалить категорию и добавить категорию. Если пользователь откроет страницу «Акции», то в ней он сможет посмотреть существующие акции, удалить акции и добавить акции. При открытие пользователем страницы «Заказы», тот сможет посмотреть все заказы, существующие в системе и их содержимое. Если пользователь откроет окно «Профиль», то в нём он сможет увидеть свои данные и сменить пароль, заполнив необходимые поля.

2.5.3 Пользовательский сценарий для повара

Если пользователь авторизуется в качестве повара, то для него откроется окно, в котором присутствует две колонки. В первой представлены заказы со статусом «Оплачено», а во второй представлены заказы, которые данный пользователь принял в готовку за текущую смену. В данном окне пользователь может просматривать заказы и их содержимое, а также изменять статусы заказов.

2.6 Прототипы основных интерфейсов

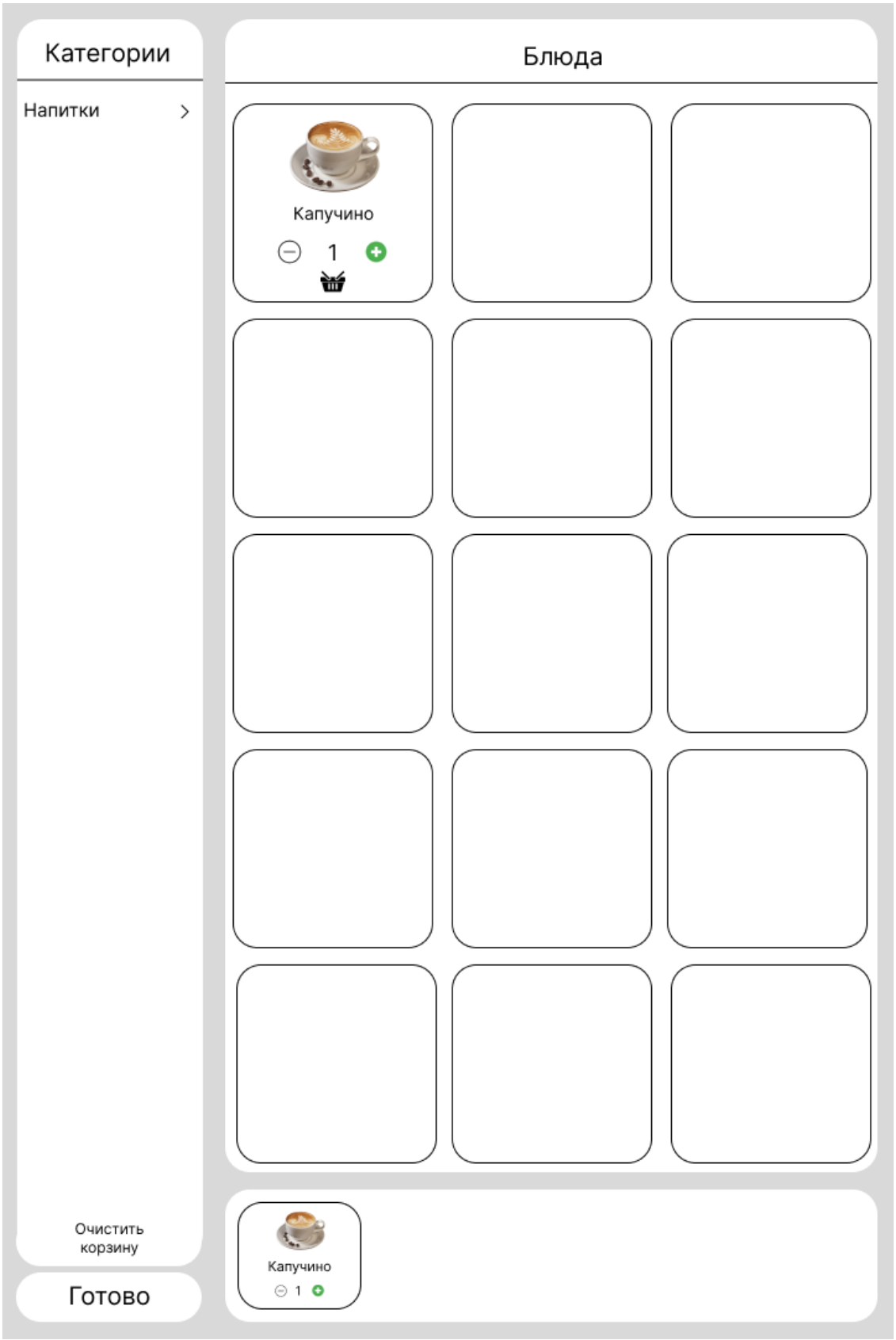


Рисунок 2.6.1 – Прототип окна покупателя

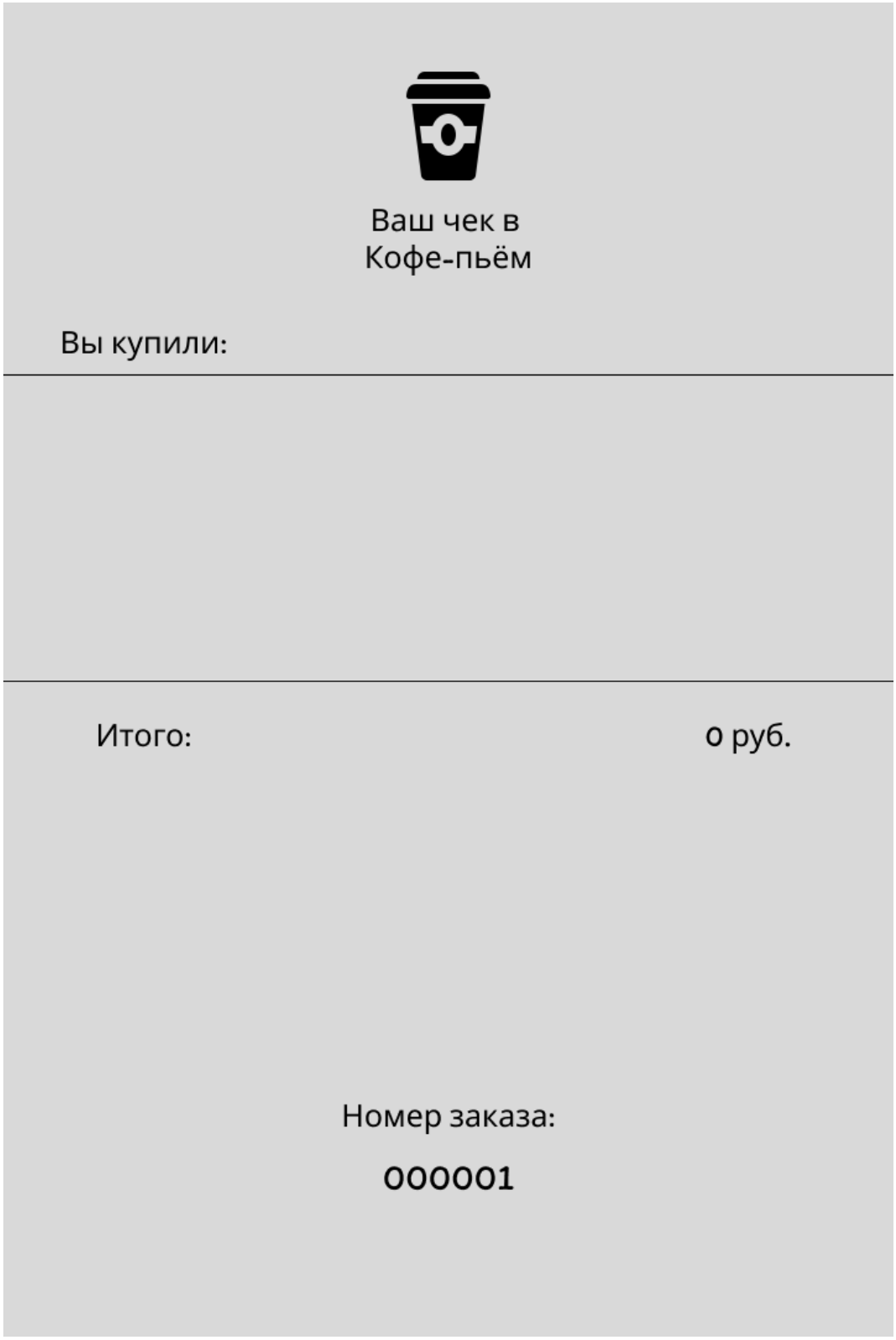


Рисунок 2.6.2 – Прототип окна «Чек»

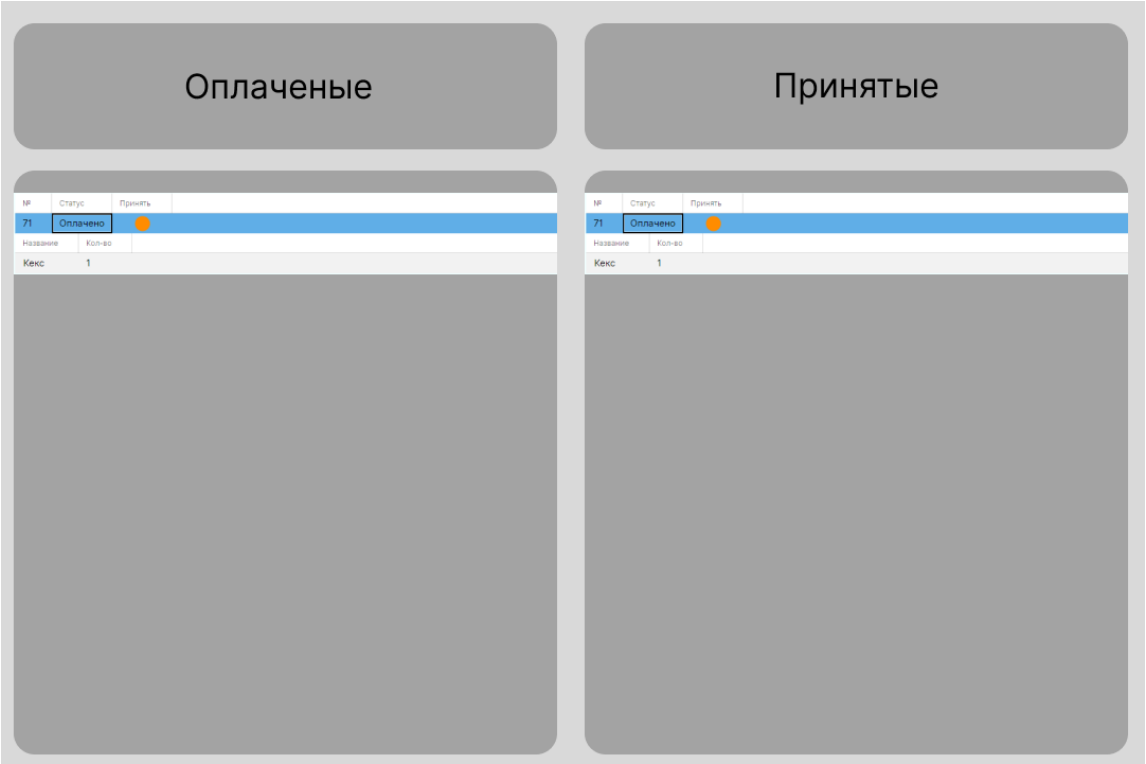


Рисунок 2.6.3 – прототип окна повара

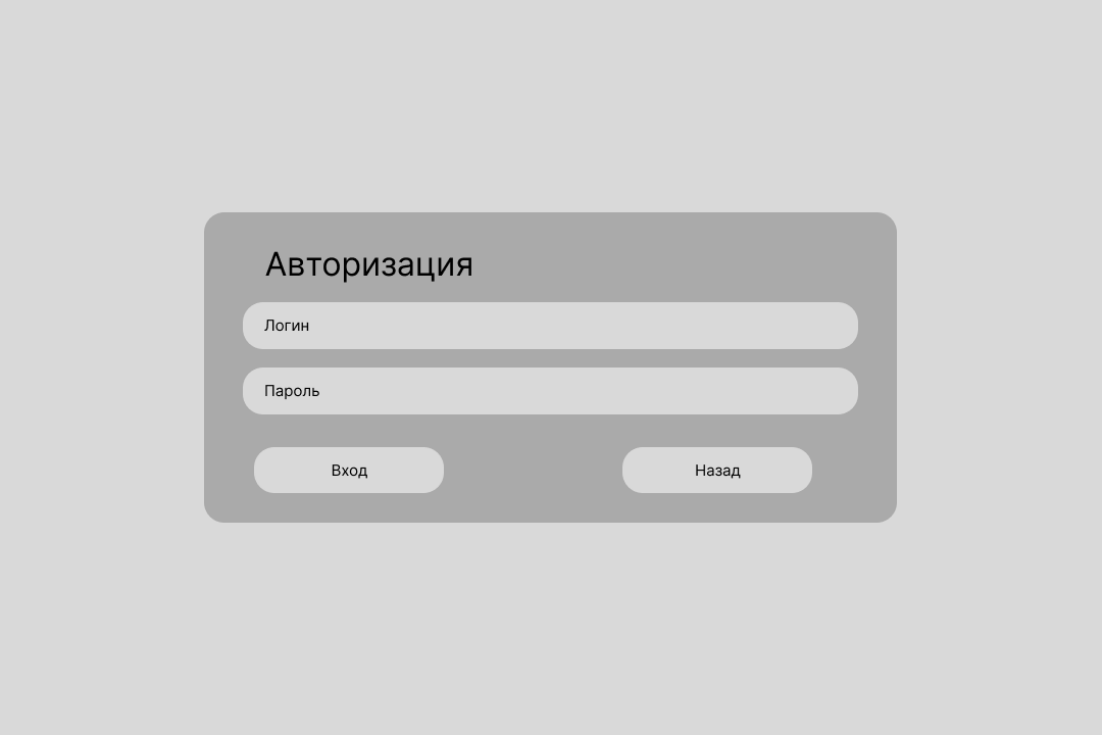


Рисунок 2.6.4 - прототип окна авторизации

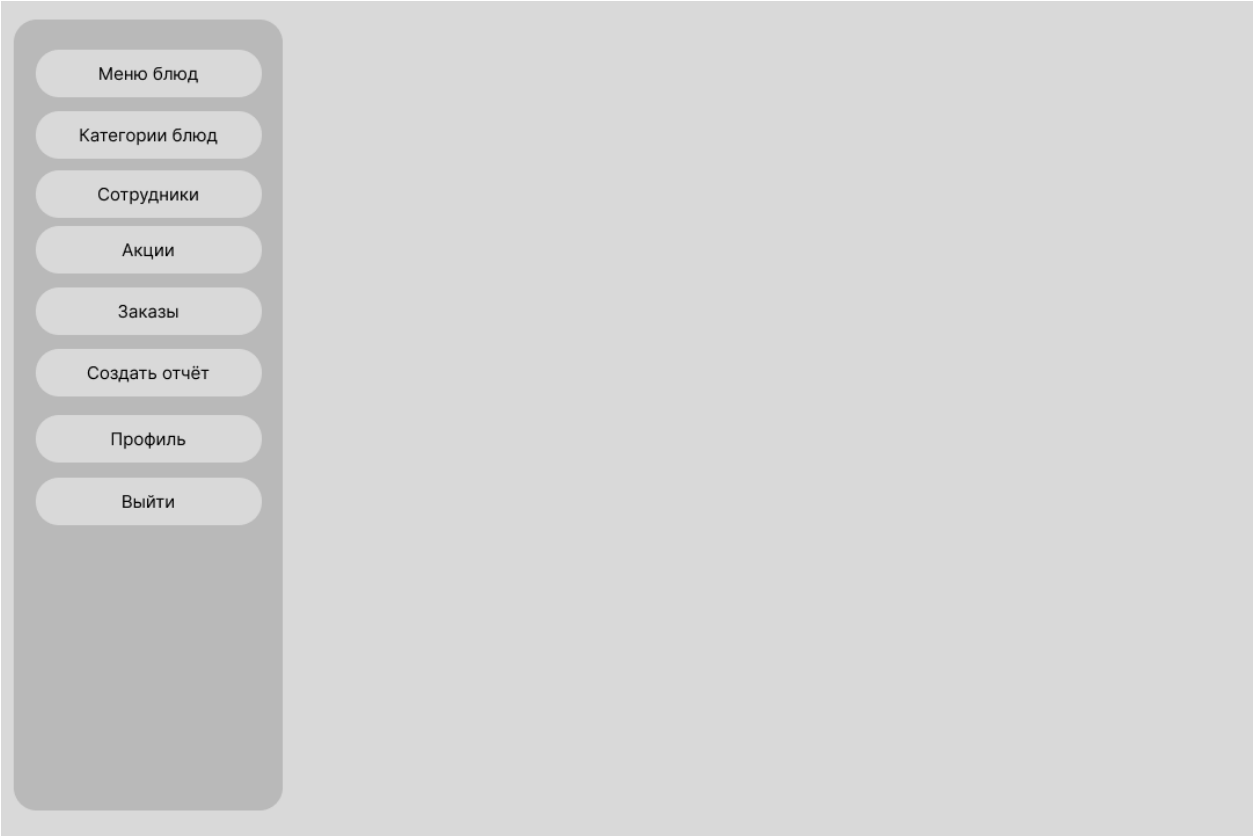


Рисунок 2.6.5 – Прототип окна администратора

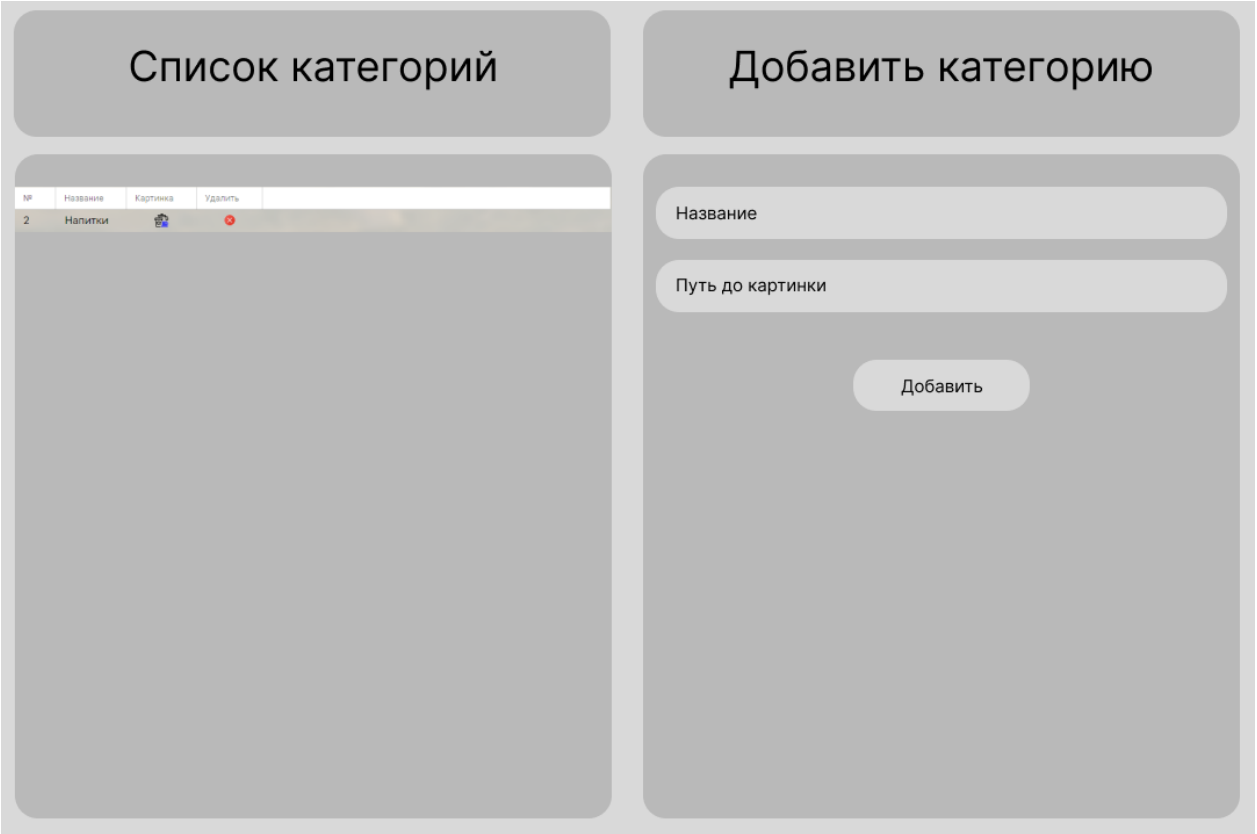


Рисунок 2.6.6 – прототип страницы категорий

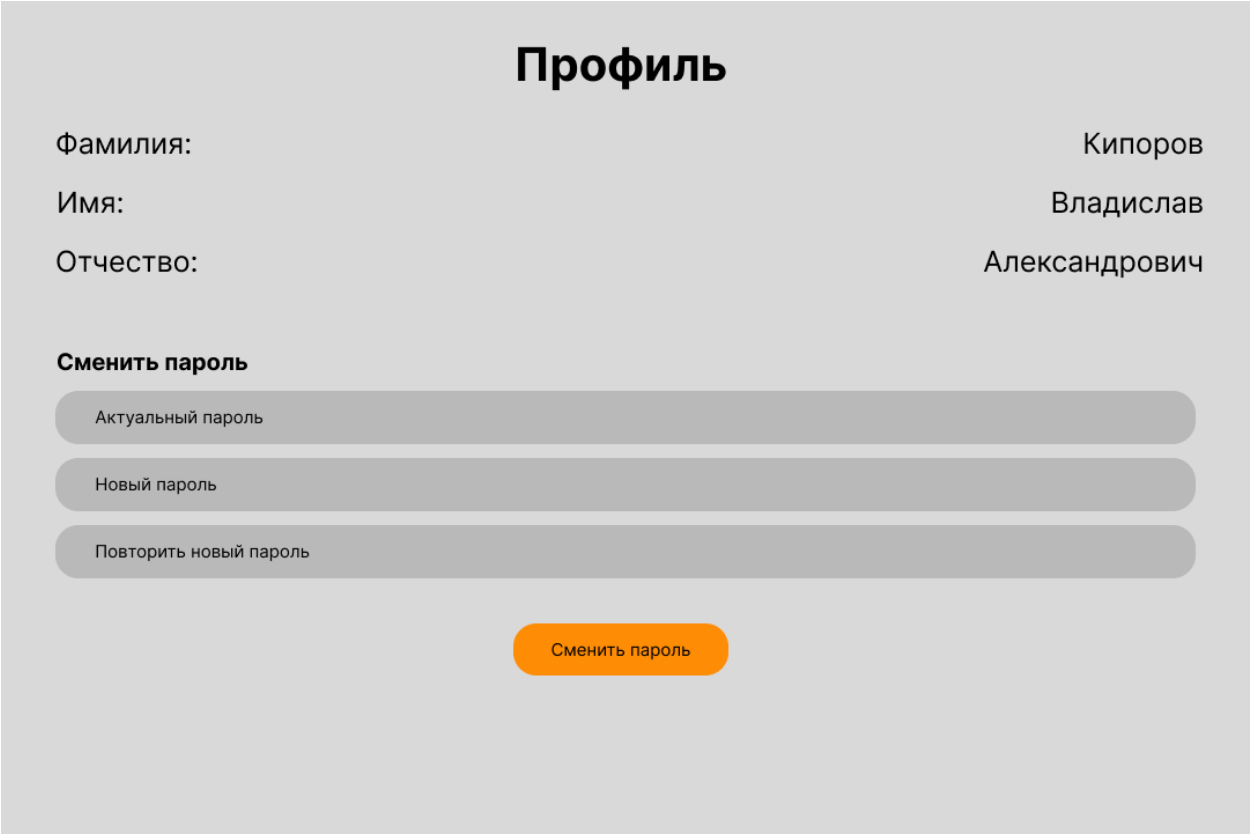


Рисунок 2.6.7 – Прототип страницы профиля

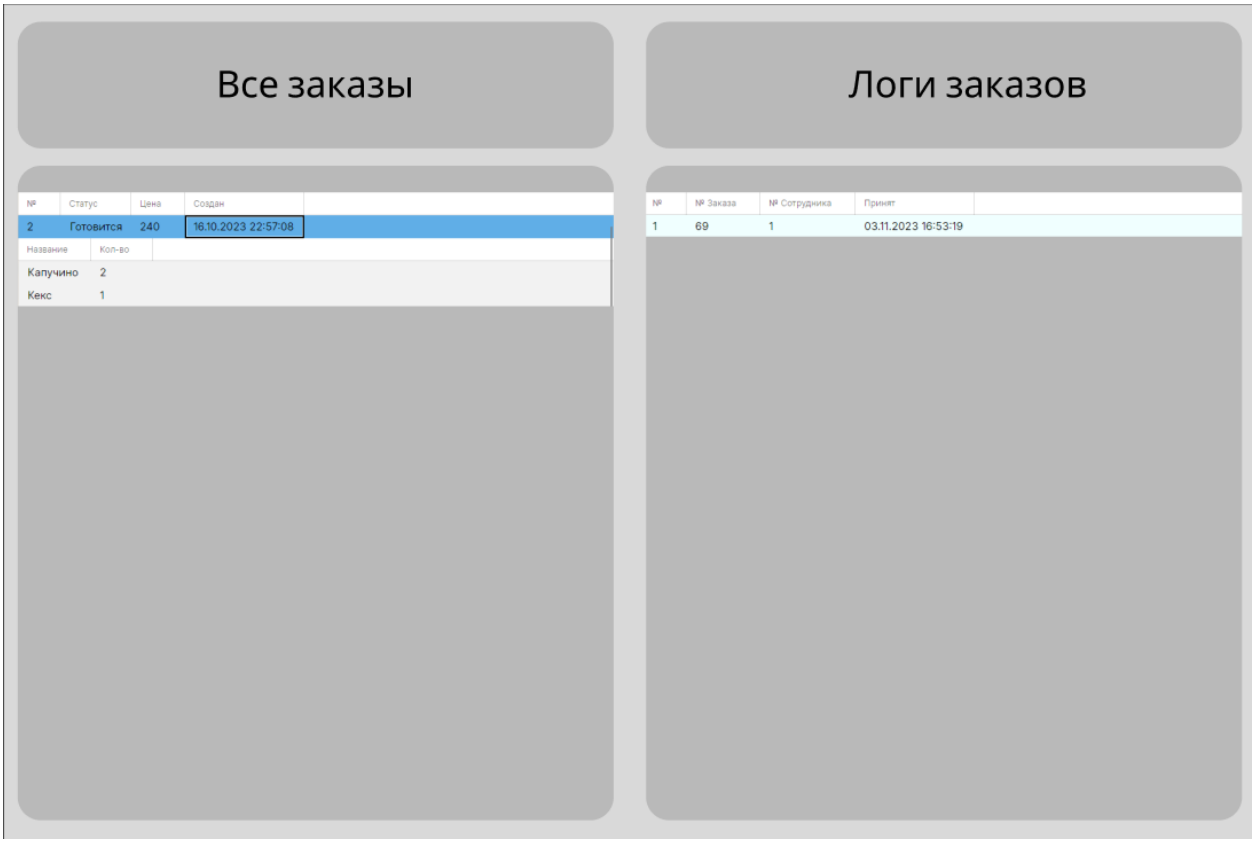


Рисунок 2.6.8 – Прототип окна заказов

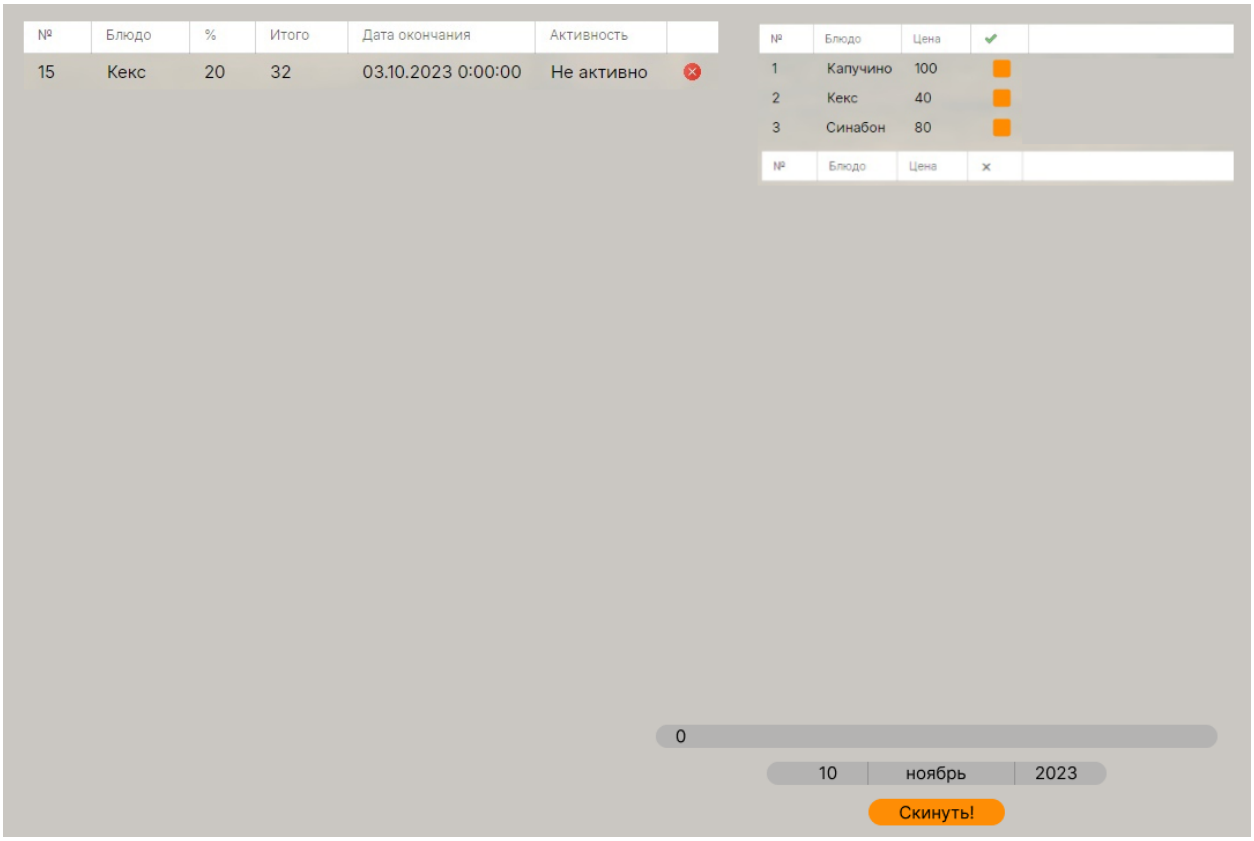


Рисунок 2.6.9 – Прототип страницы акций

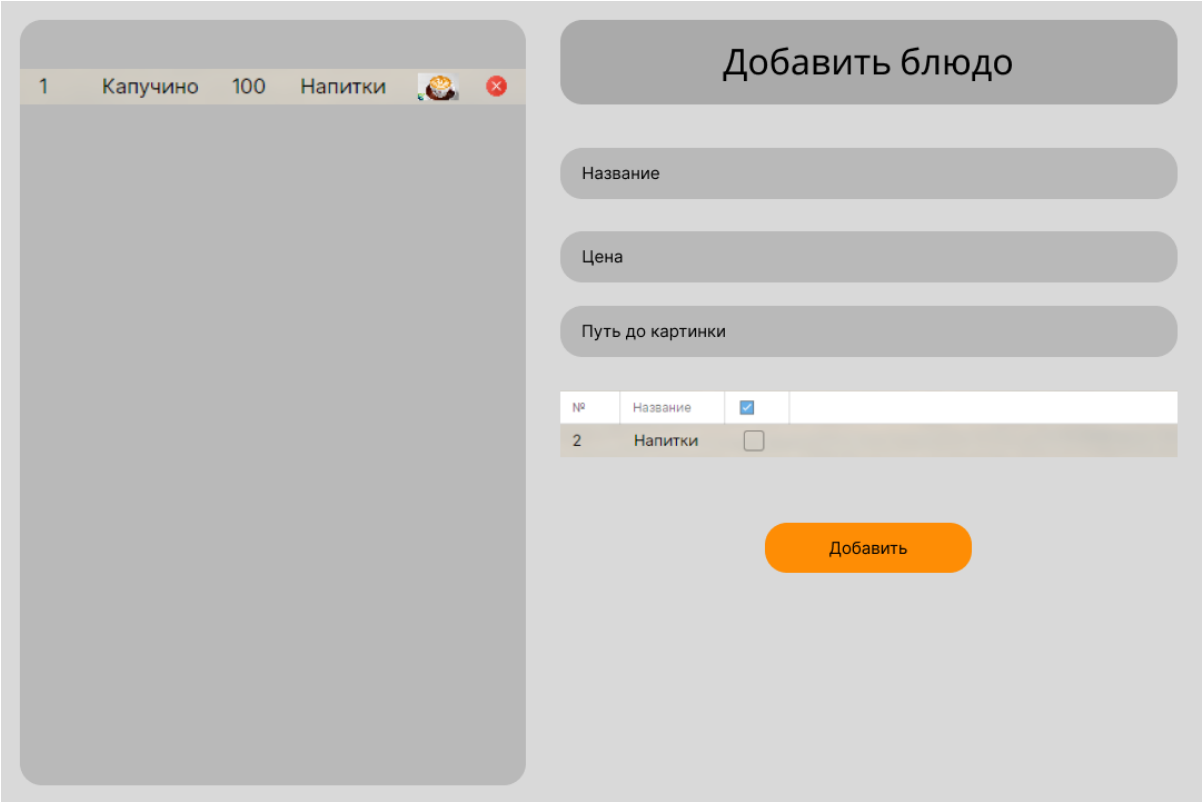


Рисунок 2.6.10 – Прототип страницы «меню блюд»

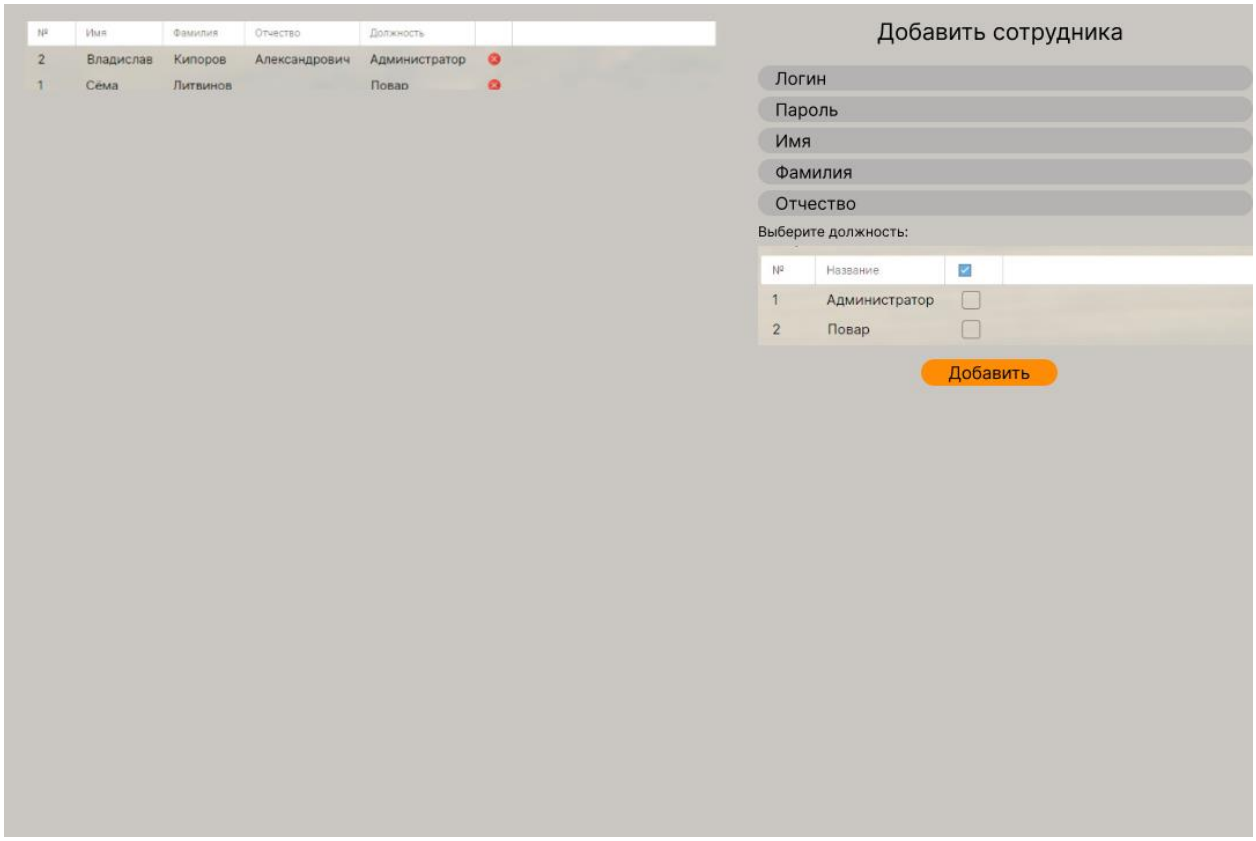


Рисунок 2.6.11 – Прототип страницы сотрудников

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта были успешно пройдены все этапы разработки информационной системы. В ходе выполнения поставленных задач была проанализирована предметная область, выбраны средства разработки, разработана и протестирована информационная система для кофейни.

В рамках системы были разработаны следующие функции:

1 Для покупателя

1.1 Выбор категории

1.2 Указание количества блюда

1.3 Добавление блюда в корзину

1.4 Удаление блюда из корзины

1.5 Очистка корзины

1.6 Создание заказа

1.7 Показ чека покупки

1.8 Система отслеживания готовности заказа

1.9 Показ акций

2 Для администратора

2.1 Авторизация

2.2 Добавление блюд, пользователей, акций и категорий

2.3 Удаление блюд, пользователей, акций и категорий

2.4 Просмотр заказов, блюд, пользователей, акций и категорий

2.5 Создание отчётов в excel-таблицу

2.6 Просмотр профиля и смена пароля

2.7 Деаутентификация

3 Для повара

3.1 Авторизация

3.2 Просмотр заказов со статусом «Оплачено»

3.3 Просмотр выполненных заказов за смену

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Avalonia UI. Документация: [Электронный ресурс]. – Режим доступа: https://avaloniaui.net

2 Microsoft. Документация: [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp

3 Metanit. Форум: [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp

4 Stack Overflow. Форум: [Электронный ресурс]. – Режим доступа: https://stackoverflow.com/questions/tagged/c%23

ПРИЛОЖЕНИЕ А. Листинг кода

А.1 Листинг кода «Окно администратора»

```
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Reactive;
using System.Windows.Input;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using Microsoft.EntityFrameworkCore;
using ReactiveUI;
using Excel = Microsoft.Office.Interop.Excel;

namespace Coffee.ViewModels;

public class AdminMainViewModel : ViewModelBase
{
    public static User AuthUserNow { get; set; }

    private ObservableCollection<Order> _order;
    public ObservableCollection<Order> Order
    {
        get => _order;
        set => this.RaiseAndSetIfChanged(ref _order, value);
    }
    private PageViewModelBase _CurrentPage;
    public PageViewModelBase CurrentPage
    {
        get { return _CurrentPage; }
        private set { this.RaiseAndSetIfChanged(ref _CurrentPage, value); }
    }
```

```
        public ReactiveCommand<Window, Unit> ExitProfile { get; }
        public ReactiveCommand<Window, Unit> CreateReport { get; }
        public ICommand OpenMenuDishPage { get; }
        public ICommand OpenEmployeesPage { get; }
        public ICommand OpenPromotionPage { get; }
        public ICommand OpenProfilePage { get; }
        public ICommand OpenCategoriesPage { get; }
        public ICommand OpenOrdersPage { get; }

        private readonly PageViewModelBase[] Pages =
        {
            new MenuPageViewModel(),
            new EmployeesPageViewModel(),
            new PromotionPageViewModel(),
            new ProfilePageViewModel(),
            new CategoriesPageViewModel(),
            new OrdersPageViewModel()
        };

        public AdminMainViewModel()
        {
            AuthUserNow = AuthorizationViewModel.AuthUser;
            ExitProfile = ReactiveCommand.Create<Window>(ExitProfileImpl);
            CreateReport = ReactiveCommand.Create<Window>(CreateReportImpl);

            Order = new ObservableCollection<Order>(Helper.GetContext().Orders.Include(x
=> x.OrderDishes).ToList());

            _CurrentPage = Pages[0];

            var canOpenMenuDish = this.WhenAnyValue(x =>
x.CurrentPage.OpenMenuDishesPage);
            OpenMenuDishPage = ReactiveCommand.Create(OpenMenuDishPageImpl,
canOpenMenuDish);
```

```
            var canOpenEmployeesPage = this.WhenAnyValue(x =>
x.CurrentPage.OpenEmployeesPage);
            OpenEmployeesPage = ReactiveCommand.Create(OpenEmployeespageImpl,
canOpenEmployeesPage);

            var canOpenPromotionPage = this.WhenAnyValue(x =>
x.CurrentPage.OpenPromotionPage);
            OpenPromotionPage = ReactiveCommand.Create(OpenPromotionPageImpl,
canOpenPromotionPage);

            var canOpenProfilePage = this.WhenAnyValue(x =>
x.CurrentPage.OpenProfilePage);
            OpenProfilePage = ReactiveCommand.Create(OpenProfilePageImpl,
canOpenProfilePage);

            var canOpenCategoriesPage = this.WhenAnyValue(x =>
x.CurrentPage.OpenCategoriesPage);
            OpenCategoriesPage = ReactiveCommand.Create(OpenCategoriesPageImpl,
canOpenCategoriesPage);

            var canOpenOrdersPage = this.WhenAnyValue(x =>
x.CurrentPage.OpenOrdersPage);
            OpenOrdersPage = ReactiveCommand.Create(OpenOrdersPageImpl,
canOpenOrdersPage);
        }

        private void OpenMenuDishPageImpl()
        {
            CurrentPage = Pages[0];
        }
        private void OpenEmployeespageImpl()
        {
            CurrentPage = Pages[1];
        }
```

```
private void OpenPromotionPageImpl()
{
    CurrentPage = Pages[2];
}

private void OpenProfilePageImpl()
{
    CurrentPage = Pages[3];
}

private void OpenCategoriesPageImpl()
{
    CurrentPage = Pages[4];
}

private void OpenOrdersPageImpl()
{
    CurrentPage = Pages[5];
}

private void ExitProfileImpl(Window obj)
{
    AuthUserNow = null;
    MainWindow mw = new MainWindow();
    mw.Show();
    obj.Close();
}

private void CreateReportImpl(Window obj)
{
     using(ExcelHelper helper = new ExcelHelper())
    {
        if (helper.Open(filePath: Path.Combine(Environment.CurrentDirectory,
@"C:\Users\V-pc\Documents\Отчёт о заказах.xlsx")))
```

```
            {
                int i = 0;
                var allOrders = Order.ToList().OrderBy(p => p.DateAndTime).ToList();
                var application = new Excel.Application();
                string[] month = new string[12] { "Январь", "Февраль", "Март", "Апрель",
"Май", "Июнь", "Июль",
                                                  "Август", "Сентябрь", "Окрябрь", "Ноябрь",
"Декабрь" };

                application.SheetsInNewWorkbook = month.Length;

                Excel.Workbook workbook = application.Workbooks.Add(Type.Missing);

                for (int j = 0; j < month.Length; ++j)
                {
                    int counter = 0;
                    int startRowIndex = 1;

                    Excel.Worksheet worksheet = application.Worksheets.Item[j + 1];
                    worksheet.Name = month[j];

                    worksheet.Cells[1][startRowIndex] = "Номер";
                    worksheet.Cells[2][startRowIndex] = "Дата";
                    worksheet.Cells[3][startRowIndex] = "Цена";

                    startRowIndex++;

                    while (allOrders.Count > i)
                    {
                        if (allOrders[i].DateAndTime.Month == j + 1)
                        {
                            worksheet.Cells[1][startRowIndex] = allOrders[i].IdOrder;
                            worksheet.Cells[2][startRowIndex] =
allOrders[i].DateAndTime.ToString("yy-MM-dd");
                            worksheet.Cells[3][startRowIndex] = allOrders[i].FullPrice;
```

```
                    counter++;
                }
                else
                {
                    break;
                }
                i++;
                startRowIndex++;
            }

            Excel.Range sumRange =
worksheet.Range[worksheet.Cells[1][startRowIndex],
                worksheet.Cells[2][startRowIndex]];
            sumRange.Merge();
            sumRange.Value = "Итого:";

            worksheet.Cells[3][startRowIndex].Formula =
                $"=SUM(C{startRowIndex - counter}:" + $"C{startRowIndex - 1}";

            worksheet.Columns.AutoFit();
            helper.Save();
        }
        application.Visible = true;
    }
  }
 }
}
```

А.2 Листинг кода «Окно авторизации»

```
using System.Linq;
using System.Reactive;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using Metsys.Bson;
```

```
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;

namespace Coffee.ViewModels;

public class AuthorizationViewModel : ViewModelBase
{
    private string _password;
    private string _login;
    public static User AuthUser { get; set; }

    private User _user;
    public User User
    {
        get => _user;
        set => this.RaiseAndSetIfChanged(ref _user, value);
    }
    public string Login
    {
        get => _login;
        set => this.RaiseAndSetIfChanged(ref _login, value);
    }
    public string Password
    {
        get => _password;
        set => this.RaiseAndSetIfChanged(ref _password, value);
    }

    public ReactiveCommand<Window, Unit> ButtonEnter { get; }
    public ReactiveCommand<Window, Unit> ButtonBack { get; }

    public AuthorizationViewModel()
    {
        ButtonEnter = ReactiveCommand.Create<Window>(OpenWindowImpl);
```

```
            ButtonBack = ReactiveCommand.Create<Window>(BackWindowImpl);
        }

        private void BackWindowImpl(Window obj)
        {
            MainWindow mv = new MainWindow();
            mv.Show();
            obj.Close();
        }

        private void OpenWindowImpl(Window obj)
        {
            User user = null;
            user = Helper.GetContext().Users.SingleOrDefault(x => x.Login == Login &
x.Password == Password);

            if (user != null)
            {
                SingIn(user, obj);
            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка", "Вы ввели
неверный логин или пароль", ButtonEnum.Ok).ShowAsync();
            }
        }

        private void SingIn(User user, Window obj)
        {
            AuthUser = user;
            if (user.IdPost == 1)
            {
                AdminMainView av = new AdminMainView();
                av.Show();
                obj.Close();
```

```
            }
            else if (user.IdPost == 2)
            {
                CookView cv = new CookView();
                cv.Show();
                obj.Close();
            }
        }
    }
```

А.3 Листинг кода «Окно чека»

```
using System.Collections.ObjectModel;
using System.Linq;
using System.Reactive;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using Metsys.Bson;
using ReactiveUI;

namespace Coffee.ViewModels;

public class CheckViewModel : ViewModelBase
{
    private ObservableCollection<Dish> _dishes;
    private ObservableCollection<OrderDish> _orderDishes;
    private ObservableCollection<Order> _orders;
    private ObservableCollection<Dish> _dishesInCheck = new
ObservableCollection<Dish>();
    private Order _lastOrder = new Order();
    private float _checkPrice;

    public ObservableCollection<Dish> Dishes
    {
        get => _dishes;
        set => this.RaiseAndSetIfChanged(ref _dishes, value);
```

```
}

public ObservableCollection<OrderDish> OrderDishes
{
   get => _orderDishes;
   set => this.RaiseAndSetIfChanged(ref _orderDishes, value);
}

public ObservableCollection<Order> Orders
{
   get => _orders;
   set => this.RaiseAndSetIfChanged(ref _orders, value);
}

public Order LastOrder
{
   get => _lastOrder;
   set => this.RaiseAndSetIfChanged(ref _lastOrder, value);
}

public ObservableCollection<Dish> DishesInCheck
{
   get => _dishesInCheck;
   set => this.RaiseAndSetIfChanged(ref _dishesInCheck, value);
}

public float CheckPrice
{
   get => _checkPrice;
   set => this.RaiseAndSetIfChanged(ref _checkPrice, value);
}

public ReactiveCommand<Window, Unit> OpenSellerWindow { get; }

public CheckViewModel()
```

```
        {
            Dishes = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            OrderDishes = new
ObservableCollection<OrderDish>(Helper.GetContext().OrderDishes.ToList());
            Orders = new ObservableCollection<Order>(Helper.GetContext().Orders.ToList());
            OpenSellerWindow =
ReactiveCommand.Create<Window>(OpenSellerWindowImpl);
            FillingDishesInCheck();
        }

        private void OpenSellerWindowImpl(Window obj)
        {
            SellerView sv = new SellerView();
            sv.Show();
            obj.Close();
        }


        private void FillingDishesInCheck()
        {
            LastOrder = Helper.GetContext().Orders.OrderByDescending(o =>
o.IdOrder).FirstOrDefault();
            CheckPrice = LastOrder.FullPrice;
            var idDishes = LastOrder.OrderDishes.Select(orderDish =>
orderDish.IdDish).ToList();

            foreach (var id in idDishes)
            {
                var edentity = Dishes.FirstOrDefault(x => x.IdDish == id);
                edentity.CountDishes = OrderDishes.
                    Where(x => x.IdDish == id && x.IdOrder == _lastOrder.IdOrder).
                    FirstOrDefault().CountDishes;
                if (edentity != null)
                {
                    _dishesInCheck.Add(new Dish()
```

```
                {
                    IdDish =  edentity.IdDish,
                    Name = edentity.Name,
                    Price = edentity.Price * edentity.CountDishes,
                    Photo = edentity.Photo,
                    CountDishes = edentity.CountDishes
                });
            }
        }
    }
}
```

А.4 Листинг кода «Окно повара»

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reactive.Linq;
using System.Threading.Tasks;
using Coffee.Context;
using Coffee.Models;
using DynamicData;
using ReactiveUI;

namespace Coffee.ViewModels;

public class CookViewModel : ViewModelBase
{
    private ObservableCollection<Order> _allOrders;
    private ObservableCollection<Dish> _dishes;
    private ObservableCollection<OrderDish> _orderDishes;
    private ObservableCollection<StatusesOrder> _statusesOrders;
    private ObservableCollection<Cooking> _cookings;
    private ObservableCollection<Cooking> _logCookings = new
ObservableCollection<Cooking>();
```

```
        private ObservableCollection<Order> _getOrder = new
ObservableCollection<Order>();
        private ObservableCollection<Order> _setOrder = new
ObservableCollection<Order>();

        public static User AuthUserNow { get; set; }

        private MyDbContext db = new MyDbContext();

        public ObservableCollection<Order> AllOrders
        {
           get => _allOrders;
           set => this.RaiseAndSetIfChanged(ref _allOrders, value);
        }

        public ObservableCollection<Dish> Dishes
        {
           get => _dishes;
           set => this.RaiseAndSetIfChanged(ref _dishes, value);
        }

        public ObservableCollection<OrderDish> OrderDishes
        {
           get => _orderDishes;
           set => this.RaiseAndSetIfChanged(ref _orderDishes, value);
        }

        public ObservableCollection<StatusesOrder> StatusesOrders
        {
           get => _statusesOrders;
           set => this.RaiseAndSetIfChanged(ref _statusesOrders, value);
        }

        private ObservableCollection<Cooking> Cookings
        {
```

```
            get => _cookings;
            set => this.RaiseAndSetIfChanged(ref _cookings, value);
        }

        private ObservableCollection<Cooking> logCookings
        {
            get => _logCookings;
            set => this.RaiseAndSetIfChanged(ref _logCookings, value);
        }

        public ObservableCollection<Order> GetOrder
        {
            get => _getOrder;
            set => this.RaiseAndSetIfChanged(ref _getOrder, value);
        }

        public ObservableCollection<Order> SetOrder
        {
            get => _setOrder;
            set => this.RaiseAndSetIfChanged(ref _setOrder, value);
        }

        public CookViewModel()
        {
            AuthUserNow = AuthorizationViewModel.AuthUser;
            AllOrders = new
ObservableCollection<Order>(Helper.GetContext().Orders.ToList());
            Dishes = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            OrderDishes = new
ObservableCollection<OrderDish>(Helper.GetContext().OrderDishes.ToList());
            StatusesOrders = new
ObservableCollection<StatusesOrder>(Helper.GetContext().StatusesOrders.ToList());
            Cookings = new
ObservableCollection<Cooking>(Helper.GetContext().Cookings.ToList());
            GetOrder.AddRange(AllOrders.Where(o => o.IdStatus == 1));
```

```
            SetOrder.AddRange(AllOrders.Where(o => (o.IdStatus != 1) &&
o.DateAndTime.Day == DateTime.Now.Day));
        }

        public void GetOrderImpl(Order order)
        {
            order.IdStatus = 2;
            GetOrder.Remove(order);
            SetOrder.Add(order);
            db.Orders.Update(order);
            db.SaveChangesAsync();
            LoggingOrders(order);
        }

        private void LoggingOrders(Order order)
        {
            Cooking cooking = new Cooking();
            cooking.IdOrder = order.IdOrder;
            cooking.IdUser = AuthUserNow.IdUser;
            cooking.DateAdmission = DateTime.Now;

            db.Cookings.AddAsync(cooking);
            db.SaveChangesAsync();
        }

        public void SetOrderImpl(Order order)
        {
            order.IdStatus = 3;
            db.Orders.Update(order);
            db.SaveChanges();
            SetOrder.Remove(order);
            SetOrder.Add(order);
            IssueOrder(order);
        }
```

```
    public async void IssueOrder(Order order)
    {
        await Task.Delay(4000);
        order.IdStatus = 4;
        db.Orders.Update(order);
        db.SaveChanges();
        SetOrder.Remove(order);
        SetOrder.Add(order);
    }
}
```

А.5 Листинг кода «Окно сотрудников»

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reactive;
using Coffee.Context;
using Coffee.Models;
using Metsys.Bson;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;

namespace Coffee.ViewModels;

public class EmployeesPageViewModel : PageViewModelBase
{
    private User _newUser = new User();
    private string _login;
    private string _password;
    private string _fName;
    private string _sName;
    private string _lName;
    private int _idPost;

    private MyDbContext db = new MyDbContext();
```

```
private ObservableCollection<User> _user;
private ObservableCollection<Post> _posts;
public ObservableCollection<User> User
{
   get => _user;
   set => this.RaiseAndSetIfChanged(ref _user, value);
}

public ObservableCollection<Post> Posts
{
   get => _posts;
   set => this.RaiseAndSetIfChanged(ref _posts, value);
}

public string Login
{
   get => _login;
   set => this.RaiseAndSetIfChanged(ref _login, value);
}

public string Password
{
   get => _password;
   set => this.RaiseAndSetIfChanged(ref _password, value);
}

public string FName
{
   get => _fName;
   set => this.RaiseAndSetIfChanged(ref _fName, value);
}

public string SName
{
```

```
    get => _sName;
    set => this.RaiseAndSetIfChanged(ref _sName, value);
}

public string LName
{
    get => _lName;
    set => this.RaiseAndSetIfChanged(ref _lName, value);
}

public int IdPost
{
    get => _idPost;
    set => this.RaiseAndSetIfChanged(ref _idPost, value);
}

private bool _OpenEmployeesPage;
public override bool OpenEmployeesPage
{
    get => _OpenEmployeesPage;
    protected set => this.RaiseAndSetIfChanged(ref _OpenEmployeesPage, value);
}

public override bool OpenMenuDishesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenPromotionPage
{
    get => true;
    protected set => throw new NotSupportedException();
}
```

```
public override bool OpenProfilePage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenCategoriesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenOrdersPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public ReactiveCommand<Unit, Unit> AddEmployee { get; }

public EmployeesPageViewModel()
{
    User = new ObservableCollection<User>(Helper.GetContext().Users.ToList());
    Posts = new ObservableCollection<Post>(Helper.GetContext().Posts.ToList());
    AddEmployee = ReactiveCommand.Create(AddEmployeeImpl);
    OpenEmployeesPage = false;
}

private void AddEmployeeImpl()
{
    var user = Helper.GetContext().Users.FirstOrDefault(x=> x.Login == Login);
    var truePost = _posts.Where(p => p.SelectPost == true).FirstOrDefault();

    if (user == null)
    {
```

```
                _newUser.Login = _login;
                _newUser.Password = _password;
                _newUser.Fname = _fName;
                _newUser.Sname = _sName;
                _newUser.Lname = _lName;
                _newUser.IdPost = truePost.IdPost;
                Helper.GetContext().Users.Add(_newUser);
                Helper.GetContext().SaveChanges();
                MessageBoxManager.GetMessageBoxStandard("Успех", "Акция добавлена",
ButtonEnum.Ok, Icon.Success).ShowAsync();
            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка", "Неверно указаны
двнные", ButtonEnum.Ok, Icon.Error).ShowAsync();
            }
        }

        public void RemoveEmployeeImpl(User u)
        {
            _user.Remove(u);
            User user = db.Users.Where(p => p.IdUser == u.IdUser).FirstOrDefault();
            db.Users.Remove(user);
            db.SaveChanges();
        }
    }
```

А.6 Листинг кода «Окно заказов и акций»

```
using System.Collections.ObjectModel;
using System.Linq;
using Coffee.Models;
using Metsys.Bson;
using ReactiveUI;

namespace Coffee.ViewModels;
```

```
public class GoListViewModel : ViewModelBase
{
    private ObservableCollection<Order> _pendingOrders = new
ObservableCollection<Order>();
    private ObservableCollection<Order> _readyOrders = new
ObservableCollection<Order>();
    private ObservableCollection<Promotion> _promotion;
    private ObservableCollection<Order> _orders;

    public ObservableCollection<Order> PendingOrder
    {
        get => _pendingOrders;
        set => this.RaiseAndSetIfChanged(ref _pendingOrders, value);
    }

    public ObservableCollection<Order> ReadyOrder
    {
        get => _readyOrders;
        set => this.RaiseAndSetIfChanged(ref _readyOrders, value);
    }

    public ObservableCollection<Promotion> Promotion
    {
        get => _promotion;
        set => this.RaiseAndSetIfChanged(ref _promotion, value);
    }

    public ObservableCollection<Order> Order
    {
        get => _orders;
        set => this.RaiseAndSetIfChanged(ref _orders, value);
    }

    public GoListViewModel()
    {
```

```
            Promotion = new
ObservableCollection<Promotion>(Helper.GetContext().Promotions.ToList());
            Order = new ObservableCollection<Order>(Helper.GetContext().Orders.ToList());
            FillingPendingOrder();
            FillingReadyOrder();
        }

        private void FillingPendingOrder()
        {
            foreach (var order in Order)
            {
                if (order.IdStatus == 3)
                {
                    _readyOrders.Add( new Order()
                    {
                        IdOrder = order.IdOrder,
                        FullPrice = order.FullPrice,
                        DateAndTime = order.DateAndTime,
                        IdStatus = order.IdStatus
                    });
                }
            }
        }

        private void FillingReadyOrder()
        {
            foreach (var order in Order)
            {
                if (order.IdStatus == 1 || order.IdStatus == 2)
                {
                    _pendingOrders.Add( new Order()
                    {
                        IdOrder = order.IdOrder,
                        FullPrice = order.FullPrice,
                        DateAndTime = order.DateAndTime,
```

```
                    IdStatus = order.IdStatus
                });
            }
        }
    }
}
```

А.7 Листинг кода «Главное окно»

```
using System.Reactive;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using ReactiveUI;

namespace Coffee.ViewModels;

public class MainWindowViewModel : ViewModelBase
{
    public ReactiveCommand<Window, Unit> Authorization { get; }
    public ReactiveCommand<Window, Unit> Seller { get; }
    public ReactiveCommand<Window, Unit> TopDishes { get; }

    public MainWindowViewModel()
    {
        Authorization =
ReactiveCommand.Create<Window>(OpenAuthorizationWindowImpl);
        Seller = ReactiveCommand.Create<Window>(OpenSellerWindowImpl);
        TopDishes = ReactiveCommand.Create<Window>(OpenTopDishesWindowImpl);
    }


    private void OpenTopDishesWindowImpl(Window obj)
    {
        GoListView glv = new GoListView();
        glv.Show();
    }
```

```
    private void OpenSellerWindowImpl(Window obj)
    {
        SellerView sv = new SellerView();
        sv.Show();
        obj.Close();
    }

    private void OpenAuthorizationWindowImpl(Window obj)
    {
        AuthorizationView av = new AuthorizationView();
        av.Show();
        obj.Close();
    }
}
```

А.8 Листинг кода «Окно блюд»

```
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Reactive;
using System.Reflection.Metadata.Ecma335;
using Avalonia.Controls;
using Avalonia.Platform.Storage;
using Coffee.Context;
using Coffee.Models;
using Metsys.Bson;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;

namespace Coffee.ViewModels;

public class MenuPageViewModel : PageViewModelBase
{
```

```
        private MyDbContext db = new MyDbContext();

        public string ImagePath;
        public string DestImagePath;
        public string ImageProgectPath;
        public string AssetsUserPath = @"C:\Users\V-
pc\Documents\учёба\RiderProjects\Coffee\Coffee\AssetsUser";

        private string _Name;
        private float _Price;

        private string _selectedImagePath;
        private bool _OpenMenuDishesPage;

        private Dish _newDish = new Dish();

        private ObservableCollection<Dish> _dishes;
        private ObservableCollection<Category> _category;
        private ObservableCollection<DishCategory> _dishCategories;

        public string Name
        {
           get => _Name;
           set => this.RaiseAndSetIfChanged(ref _Name, value);
        }

        public float Price
        {
           get => _Price;
           set => this.RaiseAndSetIfChanged(ref _Price, value);
        }


        public ObservableCollection<Dish> Dishes
        {
```

```
    get => _dishes;
    set => this.RaiseAndSetIfChanged(ref _dishes, value);
}

public ObservableCollection<Category> Category
{
    get => _category;
    set => this.RaiseAndSetIfChanged(ref _category, value);
}

public ObservableCollection<DishCategory> DishCategories
{
    get => _dishCategories;
    set => this.RaiseAndSetIfChanged(ref _dishCategories, value);
}

public string SelectedImagePath
{
    get => _selectedImagePath;
    set => this.RaiseAndSetIfChanged(ref _selectedImagePath, value);
}

public override bool OpenMenuDishesPage
{
    get => _OpenMenuDishesPage;
    protected set => this.RaiseAndSetIfChanged(ref _OpenMenuDishesPage, value);
}

public override bool OpenEmployeesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenPromotionPage
```

```
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public override bool OpenProfilePage
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public override bool OpenCategoriesPage
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public override bool OpenOrdersPage
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public ReactiveCommand<Unit, Unit> AddDish { get; }
        public ReactiveCommand<Window, Unit> SelectImge { get; }

        public MenuPageViewModel()
        {
            Dishes = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            Category = new
ObservableCollection<Category>(Helper.GetContext().Categories.ToList());
            DishCategories = new
ObservableCollection<DishCategory>(Helper.GetContext().DishCategories.ToList());

            OpenMenuDishesPage = false;
```

```
            AddDish = ReactiveCommand.Create(AddDishImpl);
            SelectImge = ReactiveCommand.Create<Window>(SelectImgeImpl);
        }

        private async void SelectImgeImpl(Window obj)
        {
            var topLevel = TopLevel.GetTopLevel(obj);

            var files = await topLevel.StorageProvider.OpenFilePickerAsync(new
FilePickerOpenOptions
            {
                Title = "Выберите изображение",
                AllowMultiple = false,
            });

            ImagePath = Convert.ToString(files[0].Path.LocalPath);
            DestImagePath = $"{AssetsUserPath}/{files[0].Name}";
            SelectedImagePath = ImagePath;
            ImageProgectPath = $"AssetsUser/{files[0].Name}";
        }

        private void AddDishImpl()
        {
            var context = Helper.GetContext();

            var dish = Helper.GetContext().Dishes.FirstOrDefault(x=> x.Name == Name);
            var selectCategory = _category.Where(c => c.SelectCategory ==
true).FirstOrDefault();
            var categories = context.Categories.
                Where(c => c.IdCategory == selectCategory.IdCategory).ToList();

            if (dish == null)
            {
                _newDish.Name = Name;
```

```
                _newDish.Price = Price;
                _newDish.Photo = ImageProgectPath;
                _newDish.DishCategories = categories.Select(x => new DishCategory()
                    { IdCategoryNavigation = x }).ToList();
                try
                {
                    File.Copy(ImagePath, DestImagePath, true);
                    Helper.GetContext().Dishes.Add(_newDish);
                    Helper.GetContext().SaveChanges();
                    Dishes.Add(_newDish);
                    MessageBoxManager.GetMessageBoxStandard("Успех", "Блюдо добавлено",
ButtonEnum.Ok, Icon.Success).ShowAsync();
                }
                catch (Exception e)
                {
                    MessageBoxManager.GetMessageBoxStandard("Ошибка", $"{e}",
ButtonEnum.Ok, Icon.Error).ShowAsync();
                }
            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка", "Неверно указаны
данные", ButtonEnum.Ok, Icon.Error).ShowAsync();
            }
        }

        public void RemoveDishImpl(DishCategory dc)
        {
            var dish = Dishes.Where(d => d.IdDish == dc.IdDish).FirstOrDefault();

            _dishCategories.Remove(dc);
            _dishes.Remove(dish);
            db.DishCategories.Remove(dc);
            db.Dishes.Remove(dish);
            db.SaveChanges();
```

```
    }
}
```

А.9 Листинг кода «Окно заказов»

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using Coffee.Models;
using ReactiveUI;

namespace Coffee.ViewModels;

public class OrdersPageViewModel : PageViewModelBase
{
    private ObservableCollection<Order> _orders;
    private ObservableCollection<Cooking> _cookings;
    private ObservableCollection<OrderDish> _orderDishes;
    private ObservableCollection<Dish> _dishes;
    private ObservableCollection<StatusesOrder> _statusesOrders;

    public ObservableCollection<Order> Orders
    {
        get => _orders;
        set => this.RaiseAndSetIfChanged(ref _orders, value);
    }

    public ObservableCollection<Cooking> Cookings
    {
        get => _cookings;
        set => this.RaiseAndSetIfChanged(ref _cookings, value);
    }

    public ObservableCollection<OrderDish> OrderDishes
    {
        get => _orderDishes;
        set => this.RaiseAndSetIfChanged(ref _orderDishes, value);
```

```
}

public ObservableCollection<Dish> Dishes
{
    get => _dishes;
    set => this.RaiseAndSetIfChanged(ref _dishes, value);
}

public ObservableCollection<StatusesOrder> StatusesOrders
{
    get => _statusesOrders;
    set => this.RaiseAndSetIfChanged(ref _statusesOrders, value);
}

private bool _openOrderPage;
public override bool OpenOrdersPage
{
    get => _openOrderPage;
    protected set => this.RaiseAndSetIfChanged(ref _openOrderPage, value);
}

public override bool OpenProfilePage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenPromotionPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenMenuDishesPage
{
```

```
            get => true;
            protected set => throw new NotSupportedException();
        }

        public override bool OpenEmployeesPage
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public override bool OpenCategoriesPage
        {
            get => true;
            protected set => throw new NotSupportedException();
        }
        public OrdersPageViewModel()
        {
            Orders = new ObservableCollection<Order>(Helper.GetContext().Orders.ToList());
            Cookings = new
ObservableCollection<Cooking>(Helper.GetContext().Cookings.ToList());
            Dishes = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            OrderDishes = new
ObservableCollection<OrderDish>(Helper.GetContext().OrderDishes.ToList());
            StatusesOrders = new
ObservableCollection<StatusesOrder>(Helper.GetContext().StatusesOrders.ToList());
        }
    }
```

А.10 Листинг кода «Окно профиля»

```
using System;
using System.Reactive;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using Metsys.Bson;
using MsBox.Avalonia;
```

```
using MsBox.Avalonia.Enums;
using ReactiveUI;

namespace Coffee.ViewModels;

public class ProfilePageViewModel : PageViewModelBase
{
    public static User _AuthUserNow { get; set; }

    private bool _OpenProfilePage;

    private string _firstpassword;
    private string _secondpassword;
    private string _oldpassword;

    public string OldPassword
    {
        get => _oldpassword;
        set => this.RaiseAndSetIfChanged(ref _oldpassword, value);
    }
    public string FirstPassword
    {
        get => _firstpassword;
        set => this.RaiseAndSetIfChanged(ref _firstpassword, value);
    }
    public string SecondPassword
    {
        get => _secondpassword;
        set => this.RaiseAndSetIfChanged(ref _secondpassword, value);
    }

    public override bool OpenProfilePage
    {
        get => _OpenProfilePage;
        protected set => this.RaiseAndSetIfChanged(ref _OpenProfilePage, value);
```

```
}

public override bool OpenPromotionPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenMenuDishesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenEmployeesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenCategoriesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenOrdersPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public ReactiveCommand<Window, Unit> ChangePassword { get; }

public ProfilePageViewModel()
```

```
        {
            _AuthUserNow = AdminMainViewModel.AuthUserNow;
            ChangePassword = ReactiveCommand.Create<Window>(ChangePasswordImpl);
        }

        private void ChangePasswordImpl(Window obj)
        {
            if (_oldpassword == AuthorizationViewModel.AuthUser.Password)
            {
                if (_oldpassword != _firstpassword)
                {
                    if (_firstpassword == _secondpassword)
                    {
                        AuthorizationView av = new AuthorizationView();
                        AuthorizationViewModel.AuthUser.Password = _firstpassword;
                        Helper.GetContext().Users.Update(AuthorizationViewModel.AuthUser);
                        Helper.GetContext().SaveChanges();
                        MessageBoxManager.GetMessageBoxStandard("Успех", "Пароль
изменён", ButtonEnum.Ok, Icon.Success).ShowWindowDialogAsync(obj);
                        av.Show();
                        // System.Threading.Thread.Sleep(5000);
                        obj.Close();
                    }
                    else
                    {
                        MessageBoxManager.GetMessageBoxStandard("Ошибка","Пароли не
совпадают",  ButtonEnum.Ok, Icon.Error).ShowWindowDialogAsync(obj);
                        return;
                    }
                }
                else
                {
                    MessageBoxManager.GetMessageBoxStandard("Ошибка","Нельзя
использовать старый пароль",  ButtonEnum.Ok, Icon.Error).ShowWindowDialogAsync(obj);
                    return;
```

```
                }

            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка","Неверный текущий
пароль",  ButtonEnum.Ok, Icon.Error).ShowWindowDialogAsync(obj);
                return;
            }
        }
    }
```

А.11 Листинг кода «Окно акций»

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reactive;
using Coffee.Context;
using Coffee.Models;
using Metsys.Bson;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;

namespace Coffee.ViewModels;

public class PromotionPageViewModel : PageViewModelBase
{
    private Promotion _newPromotion = new Promotion();
    private ObservableCollection<Promotion> Promotions = new
ObservableCollection<Promotion>();
    private ObservableCollection<Dish> _dish;
    private ObservableCollection<Promotion> _promotion;
    private ObservableCollection<Dish> _dishList = new ObservableCollection<Dish>();
    private DateTimeOffset _dateEndAction = DateTimeOffset.Now;
    private int _discountPercent = 0;
```

```
private MyDbContext db = new MyDbContext();

public int DiscountPercent
{
   get => _discountPercent;
   set => this.RaiseAndSetIfChanged(ref _discountPercent, value);
}

public DateTimeOffset DateEndAction
{
   get => _dateEndAction;
   set => this.RaiseAndSetIfChanged(ref _dateEndAction, value);
}


private bool _OpenPromotionPage;
public override bool OpenPromotionPage
{
   get => _OpenPromotionPage;
   protected set => this.RaiseAndSetIfChanged(ref _OpenPromotionPage, value);
}

public override bool OpenMenuDishesPage
{
   get => true;
   protected set => throw new NotSupportedException();
}

public override bool OpenEmployeesPage
{
   get => true;
   protected set => throw new NotSupportedException();
}

public override bool OpenProfilePage
```

```
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenCategoriesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenOrdersPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public ObservableCollection<Promotion> Promotion
{
    get => _promotion;
    set => this.RaiseAndSetIfChanged(ref _promotion, value);
}

public ObservableCollection<Dish> Dishes
{
    get => _dish;
    set => this.RaiseAndSetIfChanged(ref _dish, value);
}

public ObservableCollection<Dish> DishesLists
{
    get => _dishList;
    set => this.RaiseAndSetIfChanged(ref _dishList, value);
}
```

```
        public ReactiveCommand<Unit, Unit> AddPromotion { get; }

        public PromotionPageViewModel()
        {
            OpenPromotionPage = false;
            Promotion = new
ObservableCollection<Promotion>(Helper.GetContext().Promotions.ToList());
            Dishes = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            FillingStatus();
            AddPromotion = ReactiveCommand.Create(AddPromotionImpl);
        }

        private void FillingStatus()
        {
            for (int i = 0; i < Promotion.Count; ++i)
            {
                if (Promotion[i].FinishDate < DateTime.Now)
                {
                    Promotion[i].Activity = "Не активно";
                }
                else
                {
                    Promotion[i].Activity = "Активно";
                }
            }
        }

        private void AddPromotionImpl()
        {
            var context = Helper.GetContext();
            foreach (var d in DishesLists)
            {
                _newPromotion.IdDish = d.IdDish;
                _newPromotion.FinishDate = _dateEndAction.DateTime;
                _newPromotion.DiscountPercent = _discountPercent;
```

```
            _newPromotion.TotalPrice = d.Price - (d.Price * _discountPercent * 0.01);
            context.Promotions.Add(_newPromotion);
            context.SaveChanges();
            _newPromotion = new Promotion();
        }

        DishesLists.Clear();
        MessageBoxManager.GetMessageBoxStandard("Успех", "Акция добавлена",
ButtonEnum.Ok).ShowAsync();
    }

    public void AddDishPrePromotionImpl(Dish dish)
    {
        var edentity = _dishList.SingleOrDefault(x => x.Name == dish.Name);

        if (edentity == null)
        {
            _dishList.Add(new Dish
            {
                IdDish = dish.IdDish,
                Name = dish.Name,
                Price = dish.Price * dish.CountDishes,
                CountDishes = dish.CountDishes,
                Photo = dish.Photo
            });
        }
        else
        {
            _dishList.Remove(edentity);
            var p0 = new Dish
            {
                Name = dish.Name,
                Price = dish.Price * dish.CountDishes,
                CountDishes = dish.CountDishes,
                Photo = dish.Photo
```

```
                };
                p0.IdDish = dish.IdDish;
                _dishList.Add(p0);
            }
            var date = DateEndAction;
        }

        public void RemoveDishPrePromotionImpl(Dish dish)
        {
            _dishList.Remove(dish);
        }

        public void RemovePromotionImpl(Promotion prom)
        {
            _promotion.Remove(prom);
            Promotion promotion = db.Promotions.Where(p => p.IdPromotion ==
prom.IdPromotion).FirstOrDefault();
            db.Promotions.Remove(promotion);
            db.SaveChanges();
        }
    }
```

А.12 Листинг кода «Окно покупателя»

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reactive;
using Avalonia.Controls;
using Coffee.Models;
using Coffee.Views;
using DynamicData;
using Metsys.Bson;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;
```

```
namespace Coffee.ViewModels;

public class SellerViewModel : ViewModelBase
{
    private ObservableCollection<Dish> _dishes;
    private ObservableCollection<DishCategory> _dishCategories;
    private ObservableCollection<Category> _categories;
    private ObservableCollection<Dish> _dishesInSelectCat = new
ObservableCollection<Dish>();
    private ObservableCollection<Dish> _dishesInCart = new
ObservableCollection<Dish>();
    private static Category _selectCategory;
    private static Dish _selectedDishes;
    private static float _prePrice;

    public ObservableCollection<Dish> DishesInSelectCat
    {
        get => _dishesInSelectCat;
        set => this.RaiseAndSetIfChanged(ref _dishesInSelectCat, value);
    }

    public Category SelectCategory
    {
        get => _selectCategory;
        set
        {
            DishesInSelectCat.Clear();
            this.RaiseAndSetIfChanged(ref _selectCategory, value);

            var dishes = _dishes
                .SelectMany(d => d.DishCategories)
                .Where(x => x.IdCategory == _selectCategory.IdCategory)
                .Select(x => x.IdDishNavigation)
                .ToList();
            DishesInSelectCat.AddRange(dishes);
```

```
        }
    }

    public Dish SelectedDishes
    {
        get => _selectedDishes;
        set
        {
            this.RaiseAndSetIfChanged(ref _selectedDishes, value);

            if (SelectedDishes != null)
            {
                var edentity = _dishesInCart.SingleOrDefault(x => x.Name ==
SelectedDishes.Name);

                if (edentity == null)
                {
                    DishesInCart.Add(SelectedDishes);
                }
                else
                {
                    DishesInCart.Remove(edentity);
                    DishesInCart.Add(SelectedDishes);
                }
            }

            PrePrice = _dishesInCart.Sum(x => x.Price * x.CountDishes);
        }

    }

    public ObservableCollection<Dish> DishesInCart
    {
        get => _dishesInCart;
        set => this.RaiseAndSetIfChanged(ref _dishesInCart, value);
```

```
        }

        public ObservableCollection<Dish> Dish
        {
            get => _dishes;
            set => this.RaiseAndSetIfChanged(ref _dishes, value);
        }

        public ObservableCollection<DishCategory> DishCategory
        {
            get => _dishCategories;
            set => this.RaiseAndSetIfChanged(ref _dishCategories, value);
        }

        public ObservableCollection<Category> Category
        {
            get => _categories;
            set => this.RaiseAndSetIfChanged(ref _categories, value);
        }

        public float PrePrice
        {
            get => _prePrice;
            set => this.RaiseAndSetIfChanged(ref _prePrice, value);
        }

        public ReactiveCommand<Window, Unit> BuyButton { get; }
        public ReactiveCommand<Unit, Unit> ClearCart { get; }

        public SellerViewModel()
        {
            DishCategory = new
ObservableCollection<DishCategory>(Helper.GetContext().DishCategories.ToList());
```

```
            Category = new
ObservableCollection<Category>(Helper.GetContext().Categories.ToList());
            Dish = new ObservableCollection<Dish>(Helper.GetContext().Dishes.ToList());
            BuyButton = ReactiveCommand.Create<Window>(CreateOrderImpl);
            ClearCart = ReactiveCommand.Create(ClearCartImpl);
        }

        private void ClearAllCollection()
        {
            DishesInCart.Clear();
            PrePrice = 0;
            foreach (var d in DishesInSelectCat)
            {
                d.CountDishes = 1;
            }
        }

        private void ClearCartImpl()
        {
            DishesInCart.Clear();
        }

        private void CreateOrderImpl(Window obj)
        {
            var context = Helper.GetContext();
            var dishes = context.Dishes.
                Where(x => _dishesInCart.Select(x => x.IdDish).
                    Contains(x.IdDish)).ToList();

            if (_prePrice > 0)
            {
                Order order = new Order();
                order.DateAndTime = DateTime.Now;
                order.FullPrice = _dishesInCart.Sum(x => x.Price * x.CountDishes);
```

```
                order.OrderDishes = dishes.Select(x => new OrderDish() { IdDishNavigation = x,
CountDishes = x.CountDishes }).ToList();
                order.IdStatus = 1;
                Helper.GetContext().Orders.Add(order);
                Helper.GetContext().Orders.UpdateRange();
                Helper.GetContext().SaveChanges();

                CheckView cvw = new CheckView();
                cvw.Show();
                ClearAllCollection();
                obj.Close();
            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка", "Вы ничего не
выбрали :-(", ButtonEnum.Ok, Icon.Error).ShowAsync();
            }
        }

        public void EditCountDishImpl(Dish dish, char f)
        {
            int i = 0;

            if (f == '+')
            {
                dish.CountDishes += 1;
            }
            else
            {
                if (dish.CountDishes > 0)
                {
                    dish.CountDishes -= 1;
                }
            }
```

```
        foreach (var d in DishesInSelectCat)
        {
            if (d == dish)
            {
                break;
            }
            else
            {
                i++;
            }
        }

        DishesInSelectCat[i] = dish;
    }

    public void DeleteDishIncartImpl(Dish dish)
    {
        DishesInCart.Remove(dish);
        PrePrice = _dishesInCart.Sum(x => x.Price * x.CountDishes);
    }
}
```

А.13 Листинг кода «Окно категорий»

```
using System;
using System.Collections.ObjectModel;
using System.ComponentModel.Design;
using System.IO;
using System.Linq;
using System.Reactive;
using Avalonia.Controls;
using Avalonia.Platform.Storage;
using Coffee.Context;
using Coffee.Models;
using MsBox.Avalonia;
using MsBox.Avalonia.Enums;
using ReactiveUI;
```

```
namespace Coffee.ViewModels;

public class CategoriesPageViewModel : PageViewModelBase
{
    private ObservableCollection<Category> _categories;
    private Category _newCategory = new Category();

    private MyDbContext db = new MyDbContext();

    public string ImagePath;
    public string DestImagePath;
    public string ImageProgectPath;
    public string AssetsUserPath = @"C:\Users\V-
pc\Documents\учёба\RiderProjects\Coffee\Coffee\AssetsUser";

    private string _Name;

    private string _selectedImagePath;

    public string Name
    {
        get => _Name;
        set => this.RaiseAndSetIfChanged(ref _Name, value);
    }

    public ObservableCollection<Category> Categories
    {
        get => _categories;
        set => this.RaiseAndSetIfChanged(ref _categories, value);
    }

    public string SelectedImagePath
    {
        get => _selectedImagePath;
```

```
    set => this.RaiseAndSetIfChanged(ref _selectedImagePath, value);
}

private bool _OpenCategoriesPage;
public override bool OpenCategoriesPage
{
    get => _OpenCategoriesPage;
    protected set => this.RaiseAndSetIfChanged(ref _OpenCategoriesPage, value);
}

public override bool OpenMenuDishesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenPromotionPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenProfilePage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenEmployeesPage
{
    get => true;
    protected set => throw new NotSupportedException();
}

public override bool OpenOrdersPage
```

```
        {
            get => true;
            protected set => throw new NotSupportedException();
        }

        public ReactiveCommand<Unit, Unit> AddCategory { get; }
        public ReactiveCommand<Window, Unit> SelectImge { get; }
        public CategoriesPageViewModel()
        {
            Categories = new
ObservableCollection<Category>(Helper.GetContext().Categories.ToList());
            AddCategory = ReactiveCommand.Create(AddCategoryImpl);
            SelectImge = ReactiveCommand.Create<Window>(SelectImgeImpl);
        }
        private async void SelectImgeImpl(Window obj)
        {
            var topLevel = TopLevel.GetTopLevel(obj);

            var files = await topLevel.StorageProvider.OpenFilePickerAsync(new
FilePickerOpenOptions
            {
                Title = "Выберите изображение",
                AllowMultiple = false,
            });
            ImagePath = Convert.ToString(files[0].Path.LocalPath);
            DestImagePath = $"{AssetsUserPath}/{files[0].Name}";
            SelectedImagePath = ImagePath;
            ImageProgectPath = $"AssetsUser/{files[0].Name}";
        }

        private void AddCategoryImpl()
        {
            var context = Helper.GetContext();
            // var categories = context.Categories.
            //     Where(x => _category.Select(c => c.SelectCategory == true))
```

```
            var category = Helper.GetContext().Categories.FirstOrDefault(x=> x.Name ==
Name);
            if (category == null)
            {
                _newCategory.Name = Name;
                _newCategory.Photo = ImageProgectPath;
                try
                {
                    File.Copy(ImagePath, DestImagePath, true);
                    Helper.GetContext().Categories.Add(_newCategory);
                    Helper.GetContext().SaveChanges();
                    MessageBoxManager.GetMessageBoxStandard("Успех", "Категория
добавлена", ButtonEnum.Ok, Icon.Success).ShowAsync();
                }
                catch (Exception e)
                {
                    MessageBoxManager.GetMessageBoxStandard("Ошибка", $"{e}",
ButtonEnum.Ok, Icon.Error).ShowAsync();
                }
            }
            else
            {
                MessageBoxManager.GetMessageBoxStandard("Ошибка", "Неверно указаны
двнные", ButtonEnum.Ok, Icon.Error).ShowAsync();
            }
        }
        public void RemoveCategoryImpl(Category category)
        {
            _categories.Remove(category);
            db.Categories.Remove(category);
            db.SaveChanges();
        }
    }
```

ПРИЛОЖЕНИЕ Б. Инструкция пользователя

При запуске информационной системы пользователя встречает главное окно, на котором можно: открыть «Окно заказов и акций», нажав кнопку «Заказы и акции», открыть «Окно авторизации», нажав «Авторизоваться», и открыть «Окно покупателя», нажав кнопку «Для покупателя»

Рисунок Б.1 – Главное окно

При нажатие на кнопку «Заказы и акции», открывается отдельное окно «Окно заказов и акций» в котором можно только просматривать заказы со статусами: оплачен, принят и готовится, а так же просматривать акции.

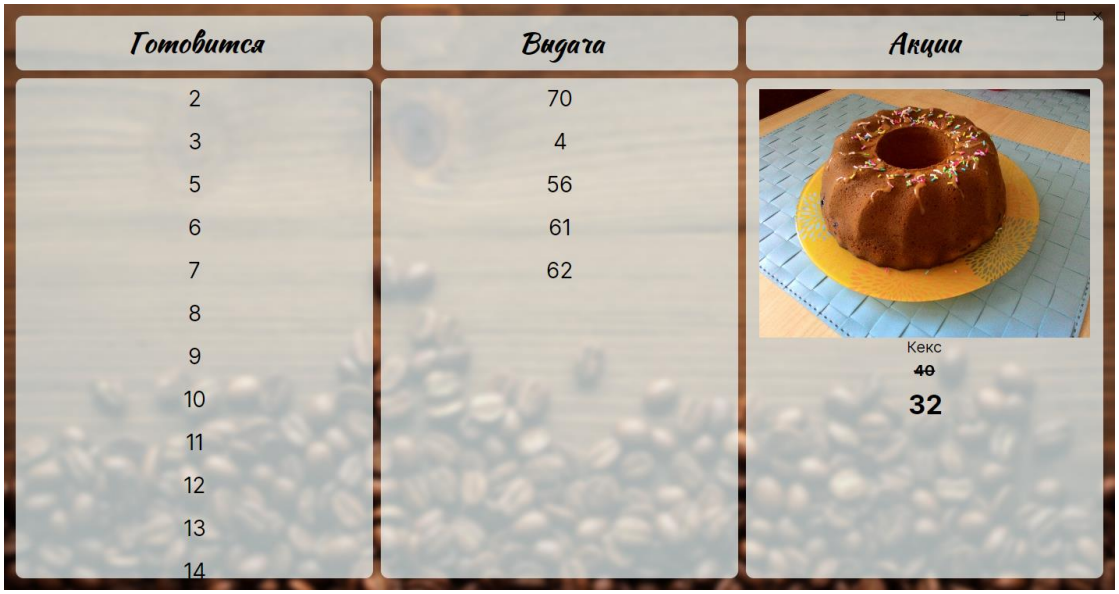


Рисунок Б.2 – Окно заказов и акций

При нажатие на кнопку «Для покупателя», открывается окно «Окно продавца» в котором можно выбрать категорию, после чего вам будут выведены блюда соответствующие выбранной категории. После чего покупатель может указать количество одинаковых блюд, нажимая на кнопки «+» и «-», и по нажатию на это блюдо добавить его в корзину. При желании покупатель может удалить блюдо из корзины нажав на кнопку «×», а также полностью отчистить корзину по нажатию кнопки «Очистить корзину». После чего пользователь может оформить заказ, нажав кнопку «Оплатить»

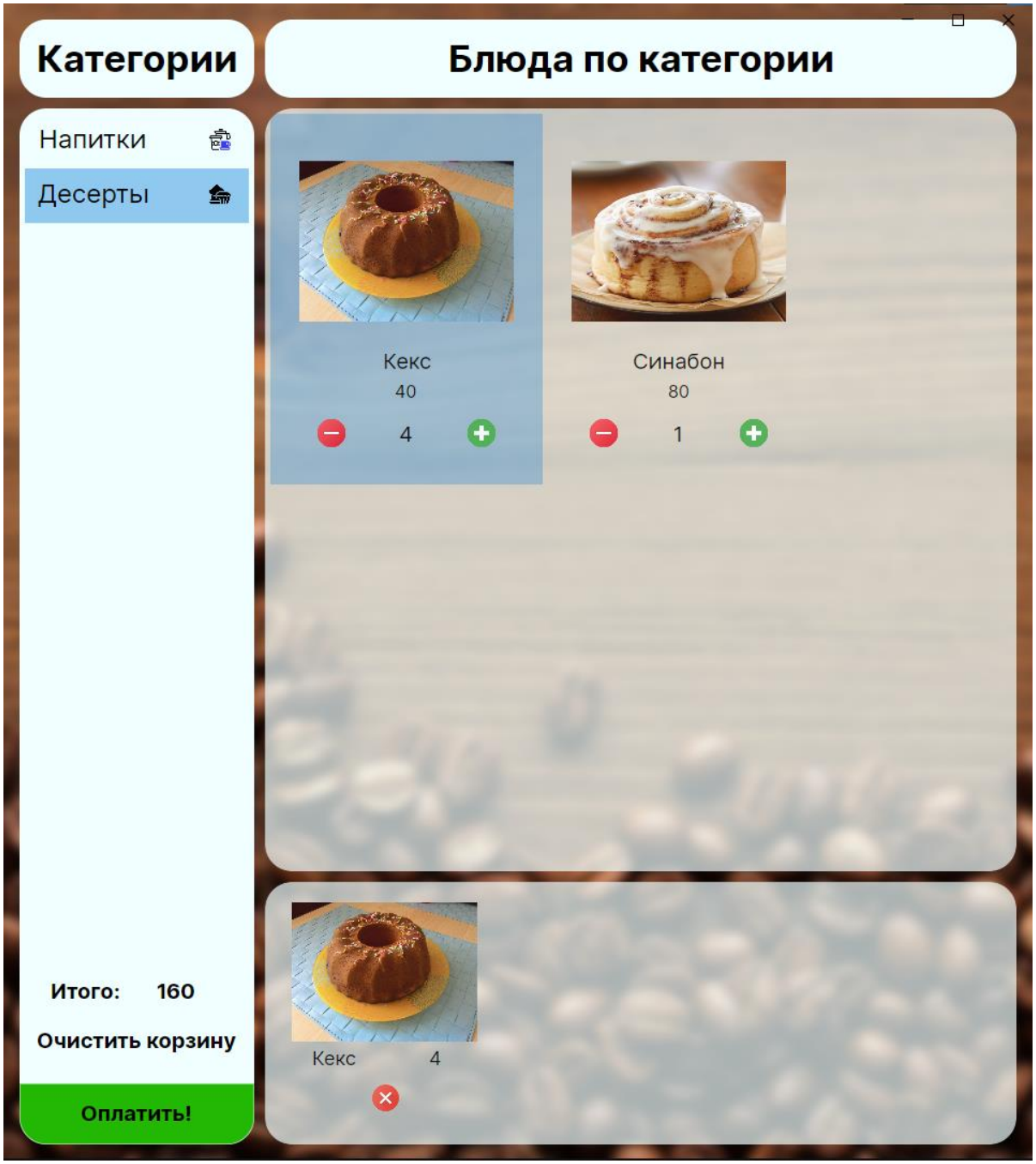


Рисунок Б.3 – Окно покупателя

После оплаты пользователю будет выведен чек с указанием его купленных блюд, итоговой стоимостью и номером заказа.

Рисунок Б.4 – Окно чека

Вернёмся к главному окну. При нажатии кнопки «Авторизоваться» появляется «Окно авторизации» в котором необходимо ввести логин и пароль, после чего необходимо нажать кнопку «Вход». При желании можно вернуться назад, нажав одноимённую кнопку «Назад».

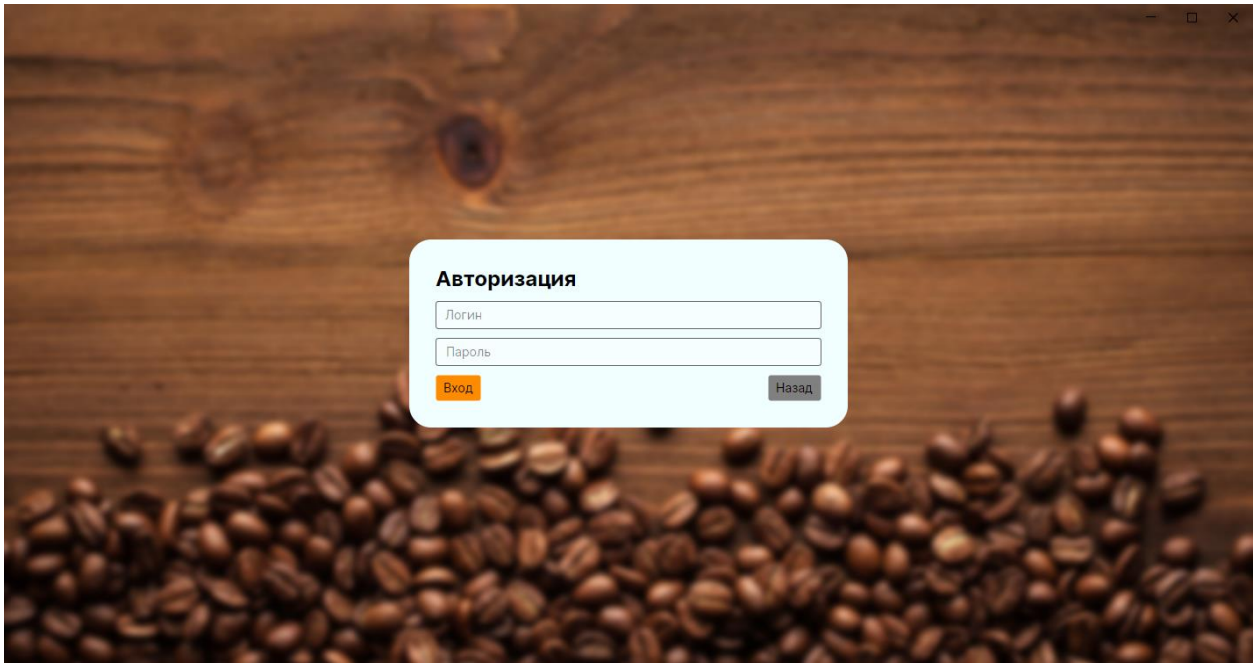


Рисунок Б.5 – Окно авторизации

В системе присутствуют роли: администратор и повар, поэтому в зависимости от роли вашей учётной записи дальнейший ход использования системы может пойти по двум сценариям.

Предположим, что вы авторизовались как повар и тогда для вас откроется «Окно повара». В нём присутствуют два столбца с заказами: оплаченными (левый столбец), содержимое которых вы можете посмотреть нажав на строку с конкретным заказом, и принять его в приготовление по нажатии на круглую кнопку, и с заказами которые вы приняли в приготовление за текущую смену (правый столбец), содержимое этих заказов вы также можете просматривать, а по нажатии на круглую кнопку заказ меняет статус с «Принят» на «Готово». После проделанных действий, по истечении 5 минут статус заказа будет автоматически изменён на «Выдано».

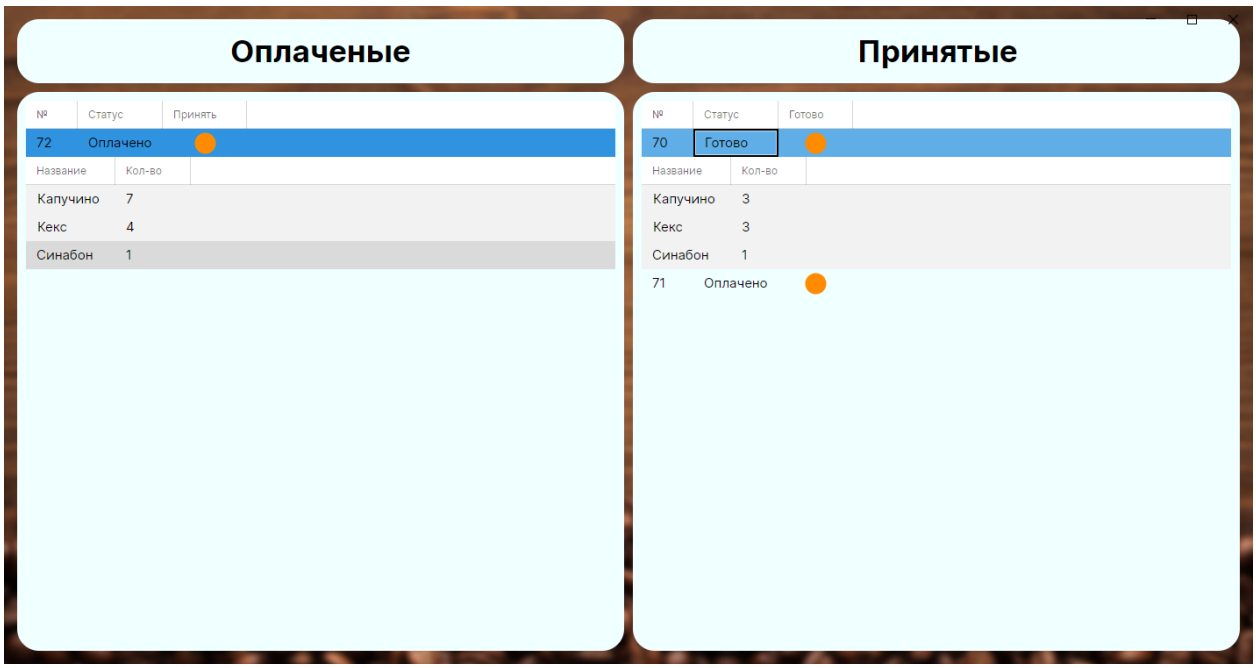


Рисунок Б.6 – Окно повара

На этом функционал для повара окончен поэтому перейдём к сценарию, в котором ваша учётная запись зарегистрирована как администратор. В этом случае вам откроется «Окно администратора», в котором предоставлено огромное количество возможностей. В левом меню вы можете выбрать с чем именно вы хотите провести операции: с блюдами, категориями, сотрудниками, акциями, заказами, просмотром своего профиля, выйти из профиля или может даже создать годовой отчёт по заказам в excel-таблицу.

Пройдёмся по порядку. При открытии этого окна вам сразу будет предоставлена страница с блюдами. В нём вы можете посмотреть какие блюда уже существуют и удалить их, а можете и добавить новое заполнив поля «Название», «Цена», выбрав категорию и указав путь до изображения блюда.

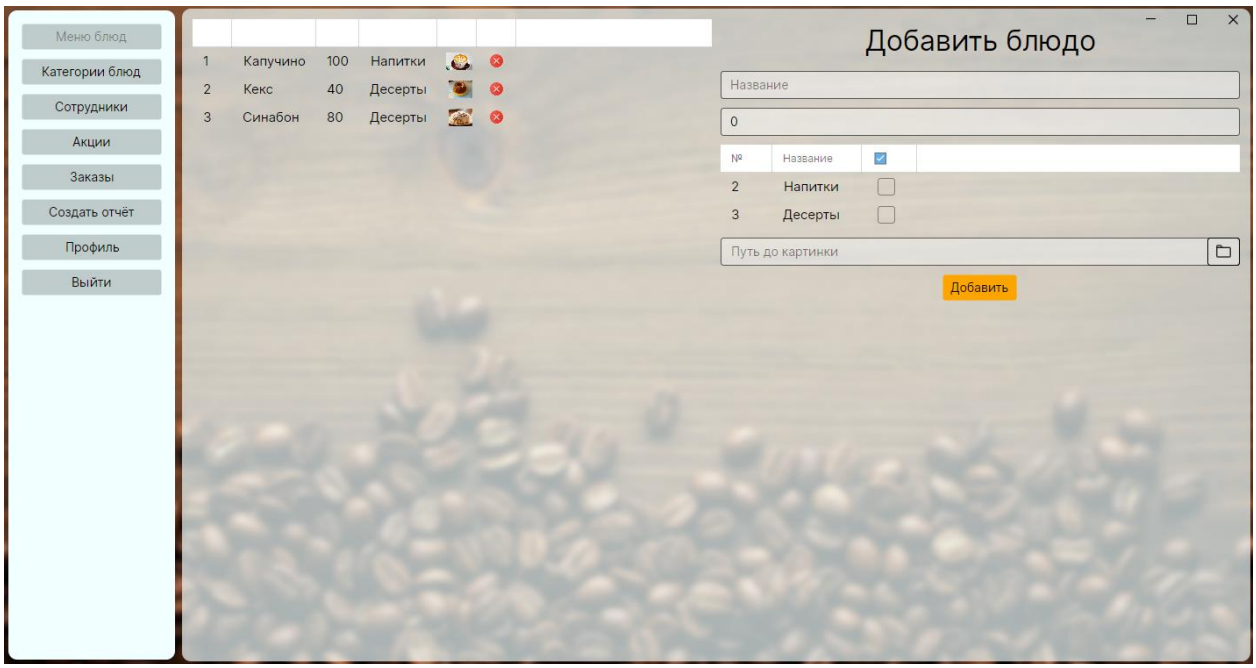


Рисунок Б.7 – Окно администратора, страница блюд

Далее, при нажатии кнопки «Категории блюд», вам откроется окно с категориями, где вы опять же можете посмотреть, удалить или добавить категорию, заполнив поля «Название» и указав путь до иллюстрации категории.

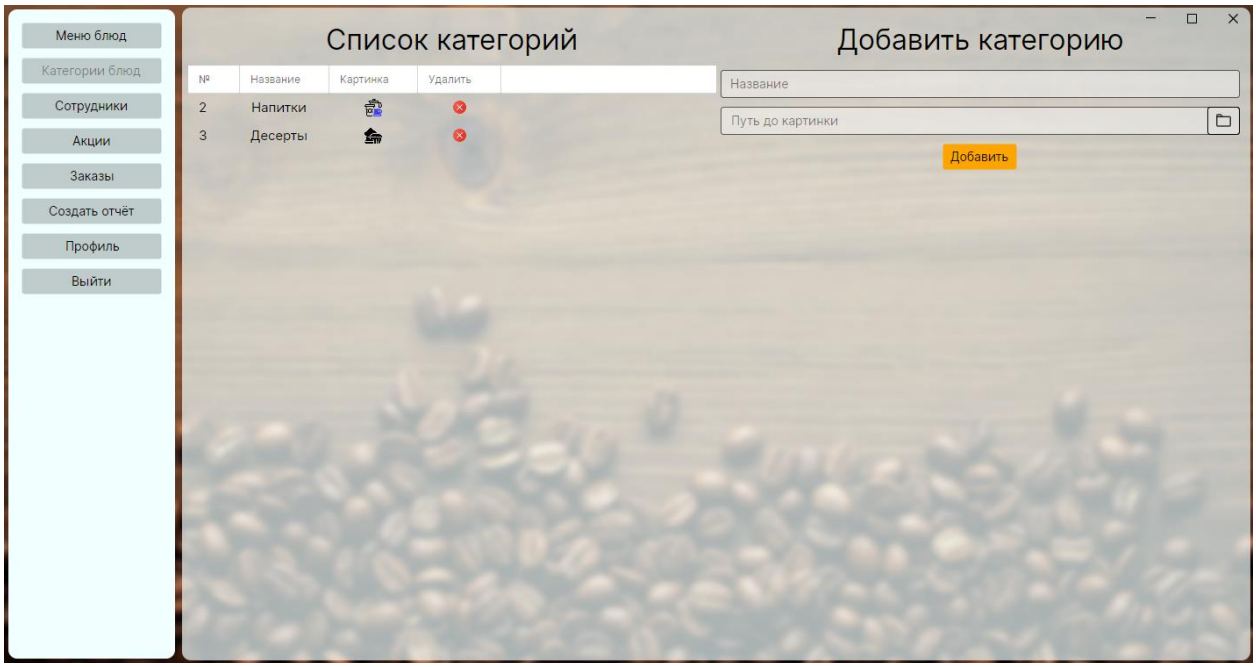


Рисунок Б.8 – Окно администратора, страница категорий

Так же вы можете в левом меню нажать на кнопку «Сотрудники» и проделать схожие операции, что и ранее: посмотреть, удалить или добавить сотрудника, заполнив поля «Логин», «Пароль», «Имя», «Фамилия»,

«Отчество» (не обязательно) и указав должность сотрудника «Администратор» или «Повар».

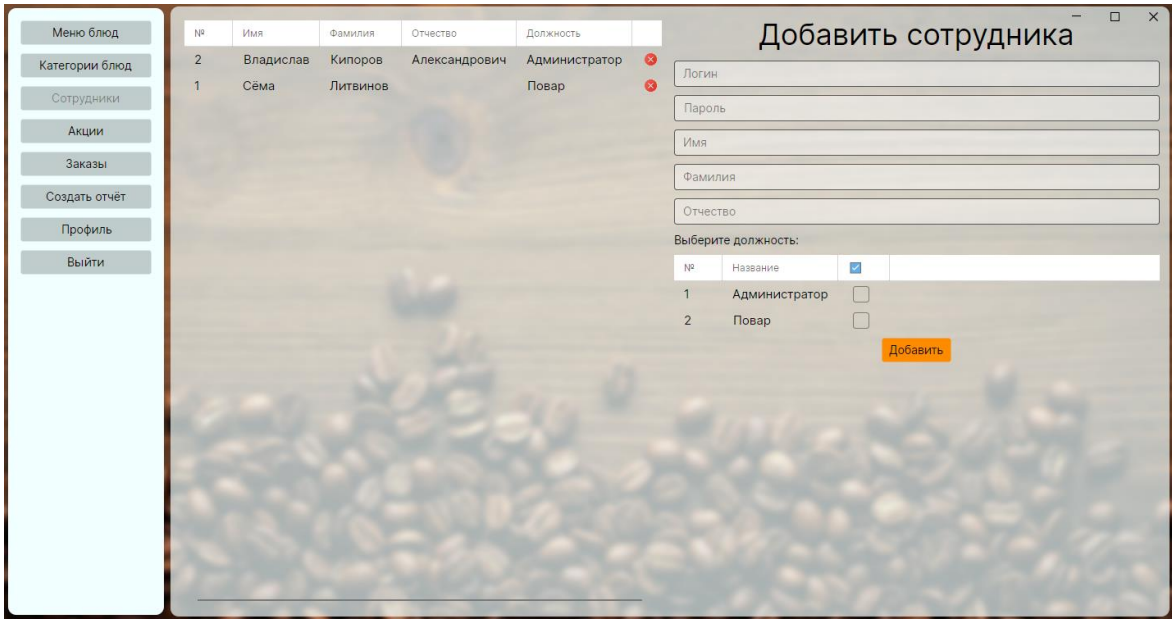


Рисунок Б.9 – Окно администратора, страница сотрудников

Если вы хотите привлечь или порадовать клиентов, то имеет смысл создать акцию на блюдо, нажав на кнопку «Акции». В этом случае вам откроется «Страница акций», где вы можете проделать знакомые вам действия: посмотреть, удалить или добавить акцию, выбрав блюда, указав процент скидки и дату, до которой акция будет активной. При достижении даты окончания акции её статус изменится на «Не активно» и перестанет показываться клиентам вашего заведения.

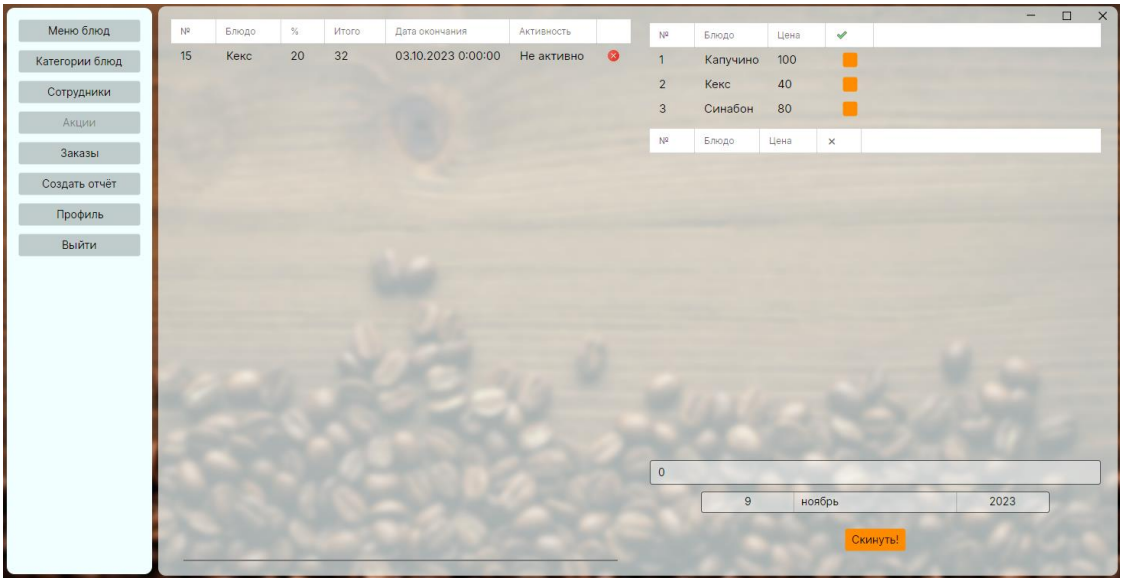
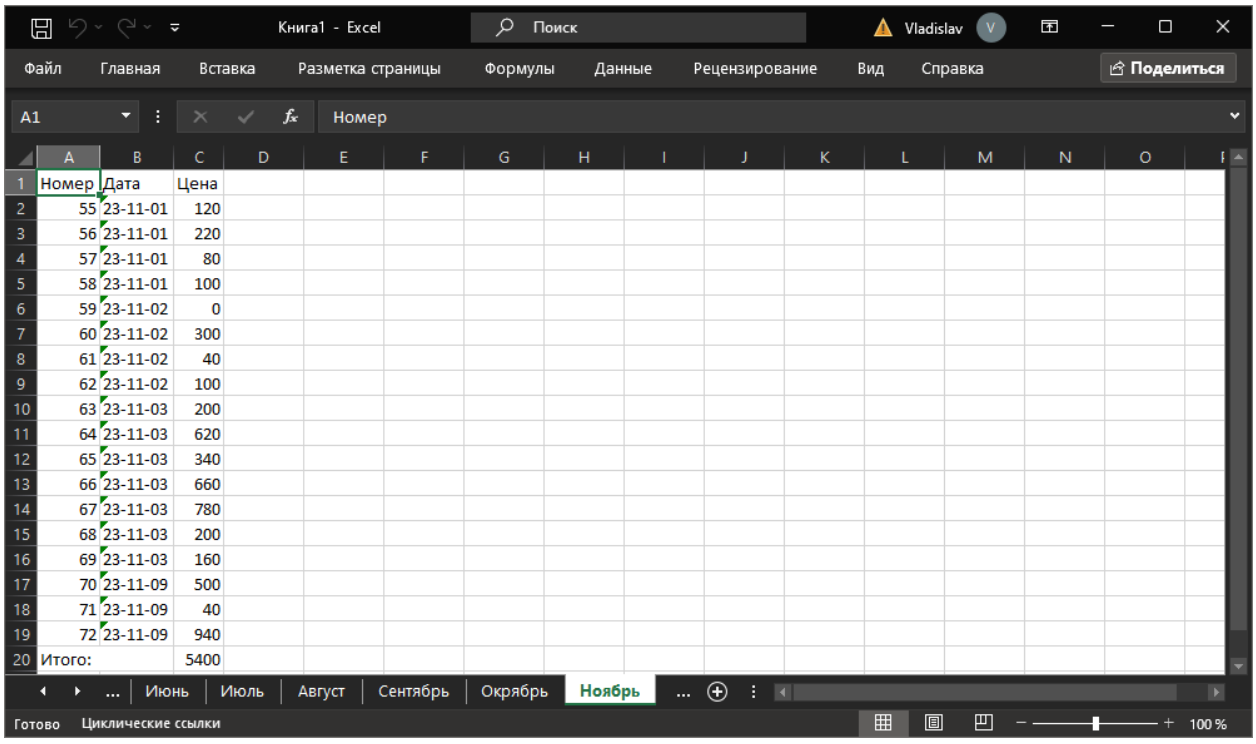


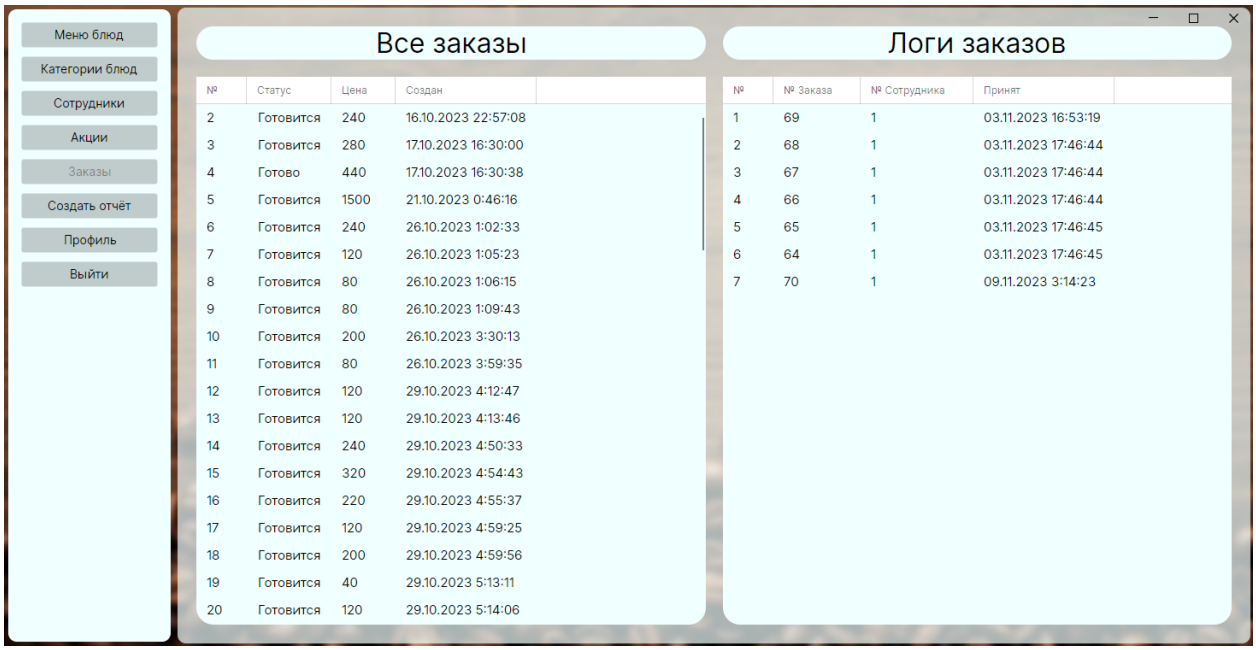
Рисунок Б.10 – Окно администратора, страница акций

Если вы решили посмотреть все свои заказы, то вы можете нажать на кнопку «Создать отчёт», после чего создастся excel-таблица с заказами, сортированными по месяцам (Рисунок Б.11), а можете нажать кнопку «Заказы», после чего откроется страница с заказами, в которой можно посмотреть заказы, их содержимое и «логи» созданные, когда повара брали заказы в приготовление (Рисунок Б.12).



	A	B	C
1	Номер	Дата	Цена
2	55	23-11-01	120
3	56	23-11-01	220
4	57	23-11-01	80
5	58	23-11-01	100
6	59	23-11-02	0
7	60	23-11-02	300
8	61	23-11-02	40
9	62	23-11-02	100
10	63	23-11-03	200
11	64	23-11-03	620
12	65	23-11-03	340
13	66	23-11-03	660
14	67	23-11-03	780
15	68	23-11-03	200
16	69	23-11-03	160
17	70	23-11-09	500
18	71	23-11-09	40
19	72	23-11-09	940
20	Итого:		5400

Рисунок Б.11 – Отчёт по заказам в excel-таблице



№	Статус	Цена	Создан
2	Готовится	240	16.10.2023 22:57:08
3	Готовится	280	17.10.2023 16:30:00
4	Готово	440	17.10.2023 16:30:38
5	Готовится	1500	21.10.2023 0:46:16
6	Готовится	240	26.10.2023 1:02:33
7	Готовится	120	26.10.2023 1:05:23
8	Готовится	80	26.10.2023 1:06:15
9	Готовится	80	26.10.2023 1:09:43
10	Готовится	200	26.10.2023 3:30:13
11	Готовится	80	26.10.2023 3:59:35
12	Готовится	120	29.10.2023 4:12:47
13	Готовится	120	29.10.2023 4:13:46
14	Готовится	240	29.10.2023 4:50:33
15	Готовится	320	29.10.2023 4:54:43
16	Готовится	220	29.10.2023 4:55:37
17	Готовится	120	29.10.2023 4:59:25
18	Готовится	200	29.10.2023 4:59:56
19	Готовится	40	29.10.2023 5:13:11
20	Готовится	120	29.10.2023 5:14:06

№	№ Заказа	№ Сотрудника	Принят
1	69	1	03.11.2023 16:53:19
2	68	1	03.11.2023 17:46:44
3	67	1	03.11.2023 17:46:44
4	66	1	03.11.2023 17:46:44
5	65	1	03.11.2023 17:46:45
6	64	1	03.11.2023 17:46:45
7	70	1	09.11.2023 3:14:23

Рисунок Б.12 – Окно администратора, страница заказов

Предпоследней кнопкой «Профиль» открывается станица профиля, в котором вы можете посмотреть свои данные и сменить пароль, заполнив поля «Актуальный пароль», «Новый пароль», «Повторите новый пароль», и нажав кнопку «Сменить пароль».

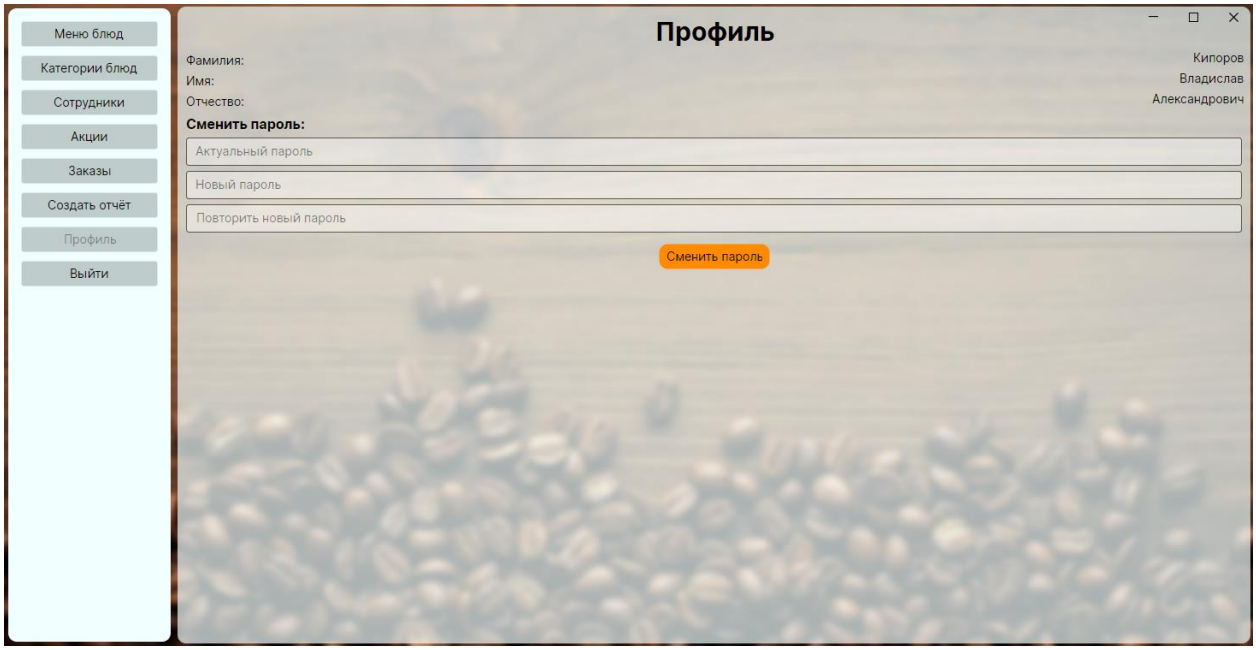


Рисунок Б.13 – Окно администратора, страница профиля

И по нажатию кнопки «Выйти» вы будете деаутентифицированы и вас вернёт на «Окно авторизации».

ПРИЛОЖЕНИЕ В. Тесты

Тестовый пример В.1

Тестовый пример#	ТП_П_В.1
Приоритет тестирования	Высокий.
Заголовок/название теста	Тестирование авторизации пользователя.
Краткое изложение теста	Проверка вводимых данных для авторизации. Условия: 1 Логин и пароль должны быть заполнены. 2 Пользователь должен быть зарегистрирован.
Этапы теста	1 Ввод необходимых данных. 2 Проверка данных на валидность.
Тестовые данные	1 Логин: 1, Пароль: 1; 2 Логин: log, Пароль: password; 3 Логин: «», Пароль: 1. 4 Логин: 1, Пароль: «»; 5 Логин: «», Пароль: «».
Ожидаемый результат	1 Пройден. 2 Не пройден. 3 Не пройден. 4 Не пройден. 5 Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно авторизации
Постусловие	Открытие окна в зависимости от роли пользователя.
Примечание/комментарии	Протестированы следующие случаи: 1 Ввод логина и пароля зарегистрированного

	пользователя. 2 Ввод несуществующего логина. 3 Пустое поле для логина. 4 Пустое поля для пароля. 5 5. Пустые оба поля.

Тестовый пример В.2

Тестовый пример#	ТП_П_В.2
Приоритет тестирования	Высокий
Заголовок/название теста	Тест добавления пользователя.
Краткое изложение теста	Проверка вводимых данных для регистрации Условия: 1 Логин, пароль, имя, фамилия и должность должны быть обязательно заполнены 2 Логины разных пользователей не должны совпадать
Этапы теста	1 Ввод необходимых данных 2 Проверка данных на валидность
Тестовые данные	1 Логин: log, пароль: pass, имя: Иван, фамилия: Иванов, отчество: Иванович, должность: Администратор. 2 Логин: «», пароль: pass, имя: Иван, фамилия: Иванов, отчество: Иванович, должность: Повар. 3 Логин: log, пароль: «», имя: Иван, фамилия: Иванов, отчество: Иванович, должность: Администратор. 4 Логин: log, пароль: pass, имя: «», фамилия: Иванов, отчество: Иванович, должность: Администратор.

	5 Логин: log, пароль: pass, имя: Иван, фамилия: «», отчество: Иванович, должность: Администратор. 6 Логин: 1, пароль: 1, имя: Иван, фамилия: Иванов, отчество: «», должность: Повар. 7 Логин: 1, пароль: 1, имя: Иван, фамилия: Иванов, отчество: Иванович, должность: «». 8 Логин: «», пароль: «», имя: Иван, фамилия: Иванов, отчество: Иванович, должность: Администратор. 9 Логин: «», пароль: «», имя: «», фамилия: Иванов, отчество: Иванович, должность: Администратор. 10 Логин: «», пароль: «», имя: «», фамилия: «», отчество: Иванович, должность: Администратор. 11 Логин: «», пароль: «», имя: «», фамилия: «», отчество: «», должность: Администратор. 12 Логин: «», пароль: «», имя: «», фамилия: «», отчество: «», должность: «». 13 Логин: 1, пароль: 1, имя: Иван, фамилия: Иванов, отчество: Иванович, должность: Администратор.
Ожидаемый результат	1 Пройден. 2 Не пройден. 3 Не пройден. 4 Не пройден. 5 Не пройден. 6 Пройден. 7 Не пройден. 8 Не пройден.

	9 Не пройден. 10 Не пройден. 11 Не пройден. 12 Не пройден. 13 Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно администратора на странице «Сотрудники»
Постусловие	Добавление нового пользователя.
Примечание/комментарии	Протестированы следующие случаи: 1 Все поля заполнены верно. 2 Заполнены все поля кроме логина. 3 Заполнены все поля кроме пароля. 4 Заполнены все поля кроме имени. 5 Заполнены все поля кроме фамилии. 6 Заполнены все поля кроме отчества. 7 Заполнены все поля кроме должности. 8 Заполнены все поля кроме логина и пароля. 9 Заполнены все поля кроме логина, пароля и имени. 10 Заполнены все поля кроме логина, пароля, имени и фамилии. 11 Заполнена только должность. 12 Все поля пусты. 13 Все поля заполнены, но пользователь с таким логином уже зарегистрирован.

Тестовый пример В.3

Тестовый пример#	ТП_П_В.3
Приоритет тестирования	Высокий
Заголовок/название теста	Тест добавления блюда.
Краткое изложение теста	Проверка вводимых данных для добавления блюда Условия: 1 Название, цена, путь до изображения и категория должны быть заполнены 2 Названия разных блюд не должны повторяться
Этапы теста	1 Ввод необходимых данных 2 Проверка данных на валидность
Тестовые данные	1 Название «Чай», цена «40», путь до изображения «E:\IMG_0002.png», категория «Напитки». 2 Название «Чай», цена «40», путь до изображения «E:\IMG_0002.png», категория «». 3 Название «Чай», цена «40», путь до изображения «», категория «Напитки». 4 Название «Чай», цена «», путь до изображения «E:\IMG_0002.png», категория «Напитки». 5 Название «», цена «40», путь до изображения «E:\ IMG_0002.png», категория «Напитки». 6 Название «Чай», цена «40», путь до изображения «E:\IMG_0002.png», категория «Напитки».
Ожидаемый результат	1 Пройден. 2 Не пройден. 3 Не пройден.

	4 Не пройден. 5 Не пройден. 6 Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно администратора на странице «Меню блюд»
Постусловие	Добавление блюда
Примечание/комментарии	Протестированы следующие случаи: 1 Все поля заполнены верно. 2 Заполнены все поля кроме категории. 3 Заполнены все поля кроме пути до изображения. 4 Заполнены все поля кроме цены. 5 Заполнены все поля кроме Названия. 6 6. Заполнены все поля, но присутствует блюдо с таким же названием.

Тестовый пример В.4

Тестовый пример#	ТП_П_В.3
Приоритет тестирования	Высокий
Заголовок/название теста	Тест добавления категории.
Краткое изложение теста	Проверка вводимых данных для добавления Условия: 1 Название и путь до изображения должны быть заполнены 2 Названия разных категорий не должны повторяться
Этапы теста	1 Ввод необходимых данных

	2 Проверка данных на валидность
Тестовые данные	1 Название «Соусы», путь до изображения «E:\IMG_2090.png» 2 Название «Соусы», путь до изображения «« 3 Название ««, путь до изображения «E:\IMG_2090.png» 4 Название «Соусы», путь до изображения «E:\IMG_2090.png»
Ожидаемый результат	1 Пройден. 2 Не пройден. 3 Не пройден. 4 Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно администратора на странице «Категории»
Постусловие	Добавление категории
Примечание/комментарии	Протестированы следующие случаи: 1 Все поля заполнены верно. 2 Заполнены все поля кроме пути до изображения. 3 Заполнены все поля кроме названия. 4 Заполнены все поля, но присутствует блюдо с таким же названием.

Тестовый пример В.5

Тестовый пример#	ТП_П_В.5
Приоритет тестирования	Высокий
Заголовок/название теста	Тест добавления акции.

Краткое изложение теста	Проверка вводимых данных для акции Условия: 3 Название и путь до изображения должны быть заполнены 4 Названия разных категорий не должны повторяться
Этапы теста	3 Ввод необходимых данных 4 Проверка данных на валидность
Тестовые данные	5 Название «Соусы», путь до изображения «E:\IMG_2090.png» 6 Название «Соусы», путь до изображения «« 7 Название ««, путь до изображения «E:\IMG_2090.png» 8 Название «Соусы», путь до изображения «E:\IMG_2090.png»
Ожидаемый результат	5 Пройден. 6 Не пройден. 7 Не пройден. 8 Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно администратора на странице «Категории»
Постусловие	Добавление категории
Примечание/комментарии	Протестированы следующие случаи: 5 Все поля заполнены верно. 6 Заполнены все поля кроме пути до изображения. 7 Заполнены все поля кроме названия.

	8 Заполнены все поля, но присутствует блюдо с таким же названием.

Тестовый пример В.6

Тестовый пример#	ТП_П_В.6
Приоритет тестирования	Низкий
Заголовок/название теста	Изменение пароля.
Краткое изложение теста	Проверка вводимых данных для смены пароля. Условия: 1. Новый пароль не должен совпадать текущим. 2. Все поля должны быть заполнены. 3. Поля «новый пароль» и «повторите пароль» должны совпадать. 4. Поле «текущий пароль» должно совпадать с паролем авторизованного пользователя.
Этапы теста	1. Открытие окна администратора на странице «Профиль» 2. Ввод необходимых данных.
Тестовые данные	1. Текущий пароль: 1, новый пароль: 12, повторите пароль: 12. 2. Текущий пароль: 1, новый пароль: 1, повторите пароль: 1. 3. Текущий пароль: «», новый пароль: 1, повторите пароль: 1. 4. Текущий пароль: 12, новый пароль: 1, повторите пароль: 1. 5. Текущий пароль: 1, новый пароль: «», повторите пароль: 12. 6. Текущий пароль: 1, новый пароль: 12,

	повторите пароль: «». 7. Текущий пароль: «», новый пароль: «», повторите пароль: «».
Ожидаемый результат	1. Пройден. 2. Пройден. 3. Не пройден. 4. Не пройден. 5. Не пройден. 6. Не пройден. 7. Не пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыто окно смены пароля «Смена пароля».
Постусловие	Изменение пароля пользователя.
Примечание/комментарии	Протестированы следующие случаи: 1. Заполнение всех полей, текущий пароль и поле «текущий пароль» совпадают, текущий пароль и поле «новый пароль» не совпадают, поле «новый пароль» и поле «повторите пароль» совпадают. 2. Заполнение всех полей, текущий пароль и поле «текущий пароль» совпадают, текущий пароль и поле «новый пароль» совпадают, поле «новый пароль» и поле «повторите пароль» совпадают. 3. Заполнение полей только «новый пароль» и «повторите пароль», поле «новый пароль» и поле «повторите пароль» совпадают. 4. Заполнение всех полей, текущий пароль и поле «текущий пароль» не совпадают, текущий пароль и поле «новый пароль» не совпадают, поле «новый пароль» и поле «повторите пароль»

	совпадают. 5. Заполнение полей только «текущий пароль» «повторите пароль», текущий пароль и поле «текущий пароль» совпадают. 6. Заполнение полей только «текущий пароль» и «новый пароль», текущий пароль и поле «текущий пароль» совпадают, текущий пароль и поле «новый пароль» не совпадают. 7. Все поля пустые.

Тестовый пример В.7

Тестовый пример#	ТП_П_В.7
Приоритет тестирования	Высокий
Заголовок/название теста	Создание заказа
Краткое изложение теста	Проверка вводимых данных для создания заказа Условия: 1. Должно быть открыто окно покупателя.
Этапы теста	1. Изменить количество блюд, если требуется. 2. Нажать на блюдо, для добавления в корзину 3. Нажать кнопку «Оплатить».
Тестовые данные	1. Номер блюда: 1, название: пончик, цена: 50, количество: 1, категория «Десерты». 2. Номер блюда: 4, название: чай, цена: 40, количество: 20, категория «Напитки»
Ожидаемый результат	1. Пройден. 2. Пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта страница покупателя.

Постусловие	Открытие окна чека
Примечание/комментарии	Протестированы следующие случаи: 1. Выбранное блюдо с параметром количества «1» добавлено. 2. Выбранное блюдо с параметром количества «20» добавлено.

Тестовый пример В.8

Тестовый пример#	ТП_П_В.8
Приоритет тестирования	Средний
Заголовок/название теста	Создание отчётов.
Краткое изложение теста	Для создания отчёта авторизуемся как администратор и нажимаем кнопку создать отчёт, после чего должен создаться excel файл.
Этапы теста	1. Авторизуемся как администратор. 2. Нажать кнопку «Создать отчёт».
Тестовые данные	Данные для авторизации: • Логин: 1. • Пароль: 1. Нажатие на кнопку «Создать отчёт».
Ожидаемый результат	Пройден.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта главная страница администратора.
Постусловие	Создание excel таблицы.
Примечание/комментарии	После создания файла необходимо сохранить его.

Тестовый пример В.9

Тестовый пример#	ТП_П_В.9

Приоритет тестирования	Средний
Заголовок/название теста	Тестирование удаления пользователя.
Краткое изложение теста	Проверка удаления пользователей. Для этого необходимо в окне администратора на странице «Сотрудники», в поле «Сотрудники» нажать на «крестик» напротив удаляемого сотрудника.
Этапы теста	1. Авторизация в роли администратора 2. Нажатие на «крестик» напротив удаляемого сотрудника.
Тестовые данные	Удаляем сотрудника с данными • №: 5. • Имя: Test. • Фамилия: Delete. • Отчество: employee.
Ожидаемый результат	Данный пользователь будет удалён.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта страница администратора.
Постусловие	Пользователь удалён.
Примечание/комментарии	Нет.

Тестовый пример В.10

Тестовый пример#	ТП_П_В.10
Приоритет тестирования	Средний
Заголовок/название теста	Тестирование удаления блюда.
Краткое изложение теста	Проверка удаления блюда. Для этого необходимо в окне администратора на странице «Меню блюд», в поле «Блюда» нажать на «крестик» напротив удаляемого блюда.

Этапы теста	1. Авторизация в роли администратора 2. Нажатие на «крестик» напротив удаляемого блюда.
Тестовые данные	Удаляем сотрудника с данными • №: 2. • Название: Синабон.
Ожидаемый результат	Выбранное блюдо будет удалено.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта страница администратора на странице «Меню блюд».
Постусловие	Блюдо удалено.
Примечание/комментарии	Нет.

Тестовый пример В.11

Тестовый пример#	ТП_П_В.11
Приоритет тестирования	Средний
Заголовок/название теста	Тестирование удаления категории.
Краткое изложение теста	Проверка удаления категории. Для этого необходимо в окне администратора на странице «Категории», в поле «Категории» нажать на «крестик» напротив удаляемой категории.
Этапы теста	1. Авторизация в роли администратора 2. Нажатие на «крестик» напротив удаляемой категории.
Тестовые данные	Удаляем категорию с данными • №: 4. • Название: Выпечка.
Ожидаемый результат	Данная категория будет удалена.

Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта страница администратора на странице «Категории».
Постусловие	Категория удалена.
Примечание/комментарии	Нет.

Тестовый пример В.12

Тестовый пример#	ТП_П_В.12
Приоритет тестирования	Средний
Заголовок/название теста	Тестирование удаления акции.
Краткое изложение теста	Проверка удаления акций. Для этого необходимо в окне администратора на странице «Акции», в поле «Акции» нажать на «крестик» напротив удаляемой акции.
Этапы теста	1. Авторизация в роли администратора 2. Нажатие на «крестик» напротив удаляемой акции.
Тестовые данные	Удаляем акцию с данными • №: 15 • Номер блюда: 1 • Процент скидки: 20
Ожидаемый результат	Данная акция будет удалена.
Фактический результат	Совпадает с ожидаемым.
Статус	Зачёт.
Предварительное условие	Открыта страница администратора на странице «Акции».
Постусловие	Акция удалена.