Автономная некоммерческая организация профессиональная образовательная организация Московский Международный Колледж Цифровых технологий "Академия ТОП"

ОТЧЕТ ПО ПРОХОЖДЕНИЮ УЧЕБНОЙ ПРАКТИКИ

Преподаватель _____ Гвоздев С.М.
личная подпись, дата

Студент _____ Борисова Д.А.
личная подпись, дата

Группа 9/3-РПО-23/1

г. Новомосковск
2025г.

Оглавление

ВВЕДЕНИЕ

В рамках прохождения учебной практики был реализован проект консольного приложения для управления ресторанов, который демонстрирует практическое применение объектно-ориентированного программирования на платформе .NET. Целью данной работы является разработка функционального программного обеспечения, которое обеспечивает комплексное управление всеми аспектами деятельности ресторана. Приложение включает функции управления меню, ингредиентами, заказами, персоналом, столами, а также система отчетности и финансового учета. В процессе разработки были использованы современные подходы к проектированию классов, организации пользовательского интерфейса консольного приложения и работе с данными. Данный отчет описывает этапы создания проекта, архитектурные решения и результаты реализации функциональных возможностей приложения.

ГЛАВА 1. СОЗДАНИЕ ПРОЕКТА

На этапе создания проекта была разработана архитектура консольного приложения для управления ресторанов, включающая множество взаимосвязанных классов, ответственных за различные аспекты работы заведения. Приложение спроектировано с использованием принципов объектно-ориентированного программирования, что обеспечивает модульность, расширяемость и удобство в поддержке кода. Основным компонентом системы является централизованный класс управления рестораном, который координирует работу всех подсистем и взаимодействует с классами, отвечающими за блюда, ингредиенты, заказы, персонал и управление столами.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Globalization;

namespace ConsoleApp3
{
    Ссылок: 8
    public class Dish
    {
        Ссылок: 10
        public int Id { get; set; }
        Ссылок: 8
        public string Name { get; set; }
        Ссылок: 7
        public string Category { get; set; }
        Ссылок: 8
        public decimal Price { get; set; }
        Ссылок: 14
        public List<IngredientPortion> Ingredients { get; set; } = new List<IngredientPortion>();

        Ссылок: 0
        public override string ToString()
        {
            return $"ID: {Id,-3} | {Name,-20} | Категория: {Category,-12} | Цена: {Price,8:C}";
        }
    }
```

Рисунок 1.1 – Создание проекта

```
Ссылок: 10
public class Ingredient
{
    Ссылок: 12
    public int Id { get; set; }
    Ссылок: 14
    public string Name { get; set; }
    Ссылок: 14
    public decimal Quantity { get; set; }
    Ссылок: 12
    public string Unit { get; set; }
    Ссылок: 11
    public decimal CostPerUnit { get; set; }

    Ссылок: 0
    public override string ToString()
    {
        return $"ID: {Id,-3} | {Name,-20} | Остаток: {Quantity,8:F2} {Unit,-4} | Цена закупки: {CostPerUnit,8:C}";
    }
}
```

Рисунок 1.2 – Создание класса с ингредиентами

```
// Связь блюда и ингредиента (сколько идёт на одну порцию)
Ссылок: 11
public class IngredientPortion
{
    Ссылок: 15
    public Ingredient Ingredient { get; set; }
    Ссылок: 13
    public decimal Amount { get; set; }

    Ссылок: 0
    public override string ToString()
    {
        return $"{Ingredient.Name} - {Amount} {Ingredient.Unit}";
    }
}
```

Рисунок 1.3 – Связь блюда и ингредиента

```
// Тип сотрудника
Ссылок: 8
public enum EmployeeRole
{
    Chef,
    Waiter,
    Manager
}
```

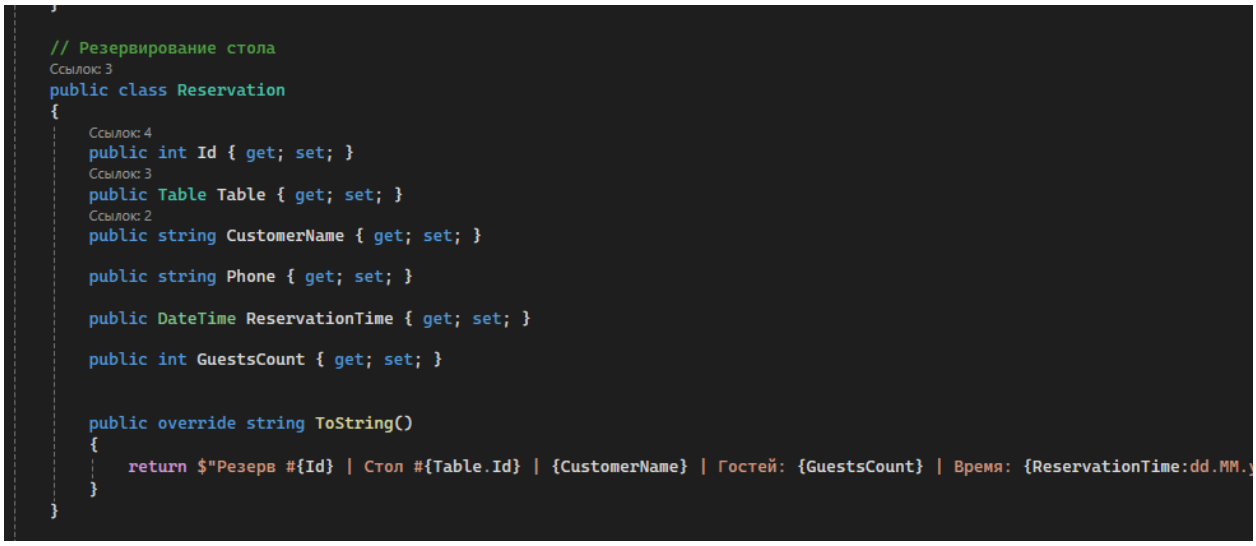
Рисунок 1.3 – Связь блюда и ингредиента

```
// Сотрудник ресторана
Ссылок: 8
public class Employee
{
    Ссылок: 9
    public int Id { get; set; }
    Ссылок: 10
    public string Name { get; set; }
    Ссылок: 10
    public EmployeeRole Role { get; set; }
    Ссылок: 9
    public decimal Salary { get; set; }

    Ссылок: 0
    public override string ToString()
    {
        return $"ID: {Id,-3} | {Name,-20} | Должность: {Role,-8} | Зарплата: {Salary,8:C}";
    }
}
```

Рисунок 1.4 – Создание класса для сотрудников ресторана

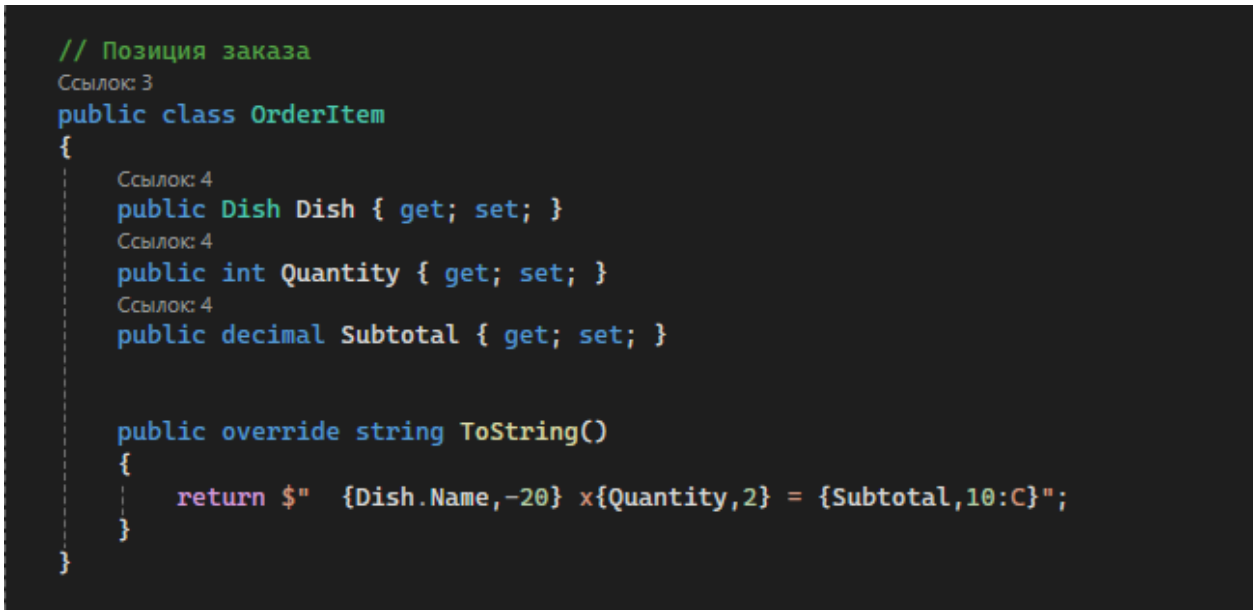
```
// Стол в зале
Ссылок: 9
public class Table
{
    Ссылок: 16
    public int Id { get; set; }
    Ссылок: 7
    public int Seats { get; set; }
    Ссылок: 6
    public bool IsActive { get; set; } = true;

    Ссылок: 0
    public override string ToString()
    {
        return $"Стол #{Id,-3} | Мест: {Seats,2} | Активен: {(IsActive ? "Да" : "Нет")}";
    }
}
```

Рисунок 1.5 – Создание класса для столов в зале



```
// Резервирование стола
Ссылок: 3
public class Reservation
{
    Ссылок: 4
    public int Id { get; set; }
    Ссылок: 3
    public Table Table { get; set; }
    Ссылок: 2
    public string CustomerName { get; set; }

    public string Phone { get; set; }

    public DateTime ReservationTime { get; set; }

    public int GuestsCount { get; set; }


    public override string ToString()
    {
        return $"Резерв #{Id} | Стол #{Table.Id} | {CustomerName} | Гостей: {GuestsCount} | Время: {ReservationTime:dd.MM.
    }
}
```

Рисунок 1.6 – Создание класса для резервирования стола

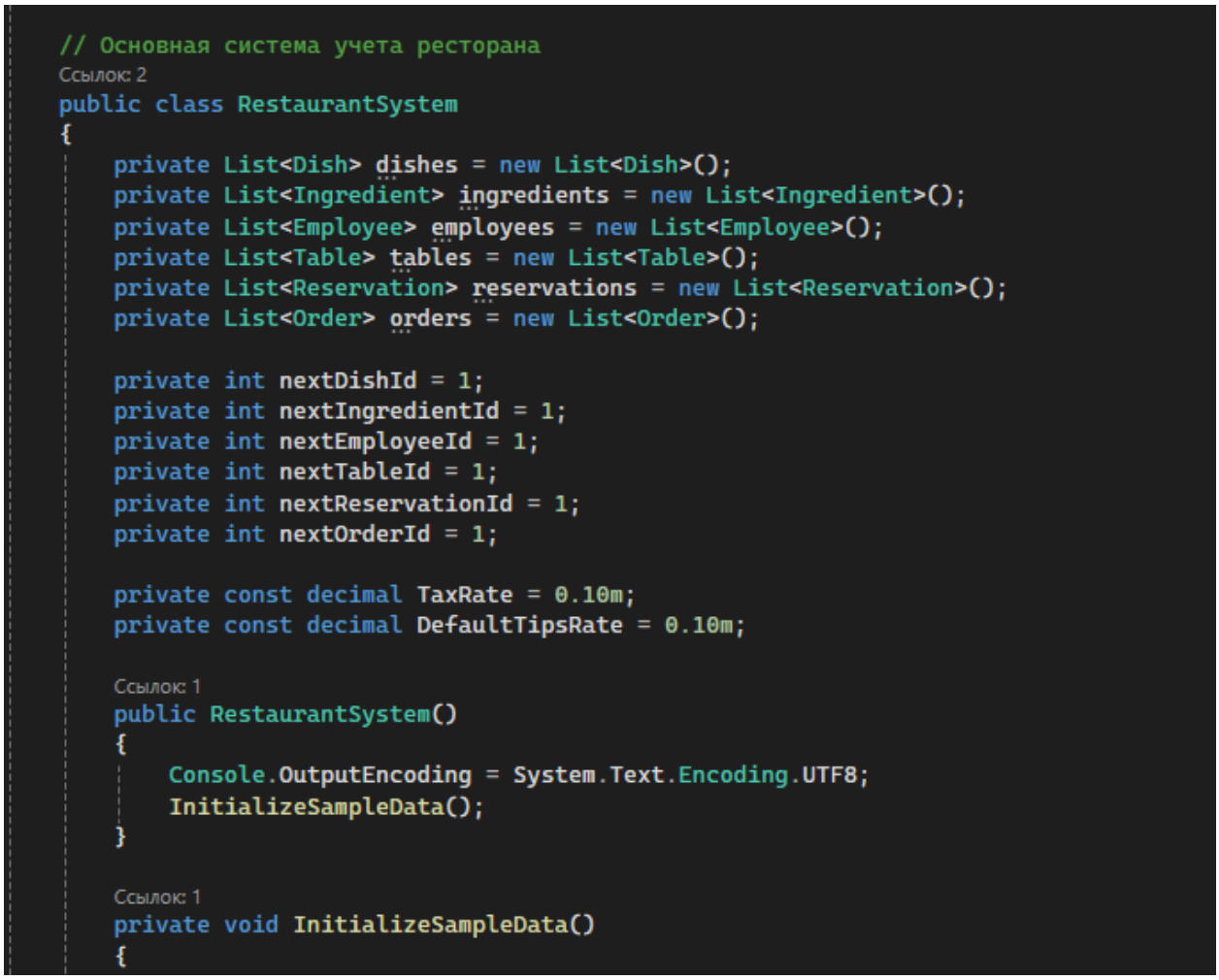


```
// Позиция заказа
Ссылок: 3
public class OrderItem
{
    Ссылок: 4
    public Dish Dish { get; set; }
    Ссылок: 4
    public int Quantity { get; set; }
    Ссылок: 4
    public decimal Subtotal { get; set; }


    public override string ToString()
    {
        return $"  {Dish.Name,-20} x{Quantity,2} = {Subtotal,10:C}";
    }
}
```

Рисунок 1.7 – Создание класса для позиции заказа

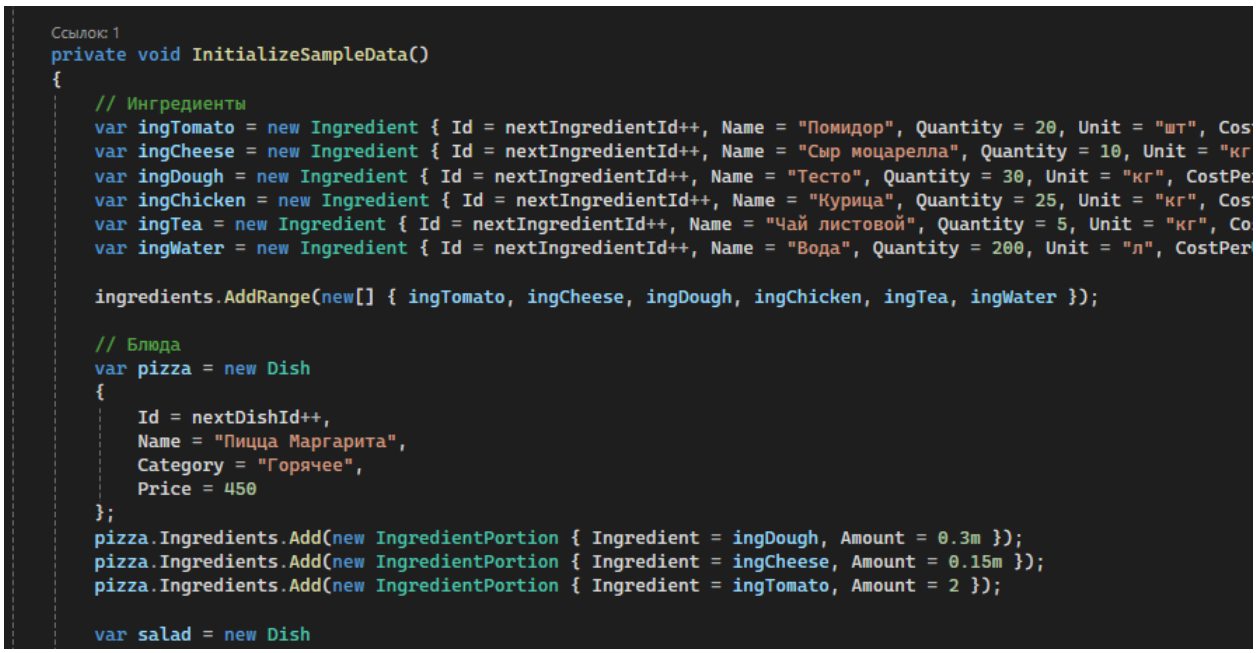
```
// Заказ со стола
Ссылок: 5
public class Order
{
    Ссылок: 5
    public int Id { get; set; }
    Ссылок: 2
    public Table Table { get; set; }
    Ссылок: 3
    public Employee Waiter { get; set; }
    Ссылок: 4
    public DateTime OrderTime { get; set; }
    Ссылок: 7
    public List<OrderItem> Items { get; set; } = new List<OrderItem>();

    public decimal Subtotal { get; set; }
    Ссылок: 5
    public decimal Tax { get; set; }

    public decimal Tips { get; set; }

    public decimal Total { get; set; }

    public bool IsClosed { get; set; }


    public override string ToString()
    {
        return $"Заказ #{Id} | Стол #{Table.Id} | Официант: {Waiter.Name} | Время: {OrderTime:HH:mm} | Сумма: {Total,8
    }
}
```

Рисунок 1.8 – Создание класса заказа со стола



```
// Основная система учета ресторана
Ссылок: 2
public class RestaurantSystem
{
    private List<Dish> dishes = new List<Dish>();
    private List<Ingredient> ingredients = new List<Ingredient>();
    private List<Employee> employees = new List<Employee>();
    private List<Table> tables = new List<Table>();
    private List<Reservation> reservations = new List<Reservation>();
    private List<Order> orders = new List<Order>();

    private int nextDishId = 1;
    private int nextIngredientId = 1;
    private int nextEmployeeId = 1;
    private int nextTableId = 1;
    private int nextReservationId = 1;
    private int nextOrderId = 1;

    private const decimal TaxRate = 0.10m;
    private const decimal DefaultTipsRate = 0.10m;

    Ссылок: 1
    public RestaurantSystem()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        InitializeSampleData();
    }

    Ссылок: 1
    private void InitializeSampleData()
    {
```

Рисунок 1.9 – Создание класса для системы учета ресторана

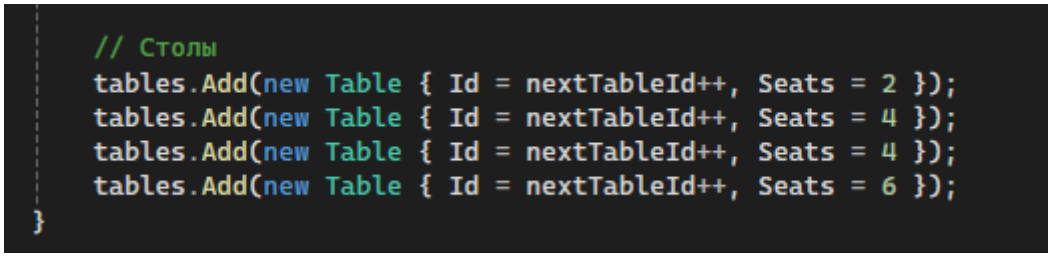


```
Ссылок: 1
private void InitializeSampleData()
{
    // Ингредиенты
    var ingTomato = new Ingredient { Id = nextIngredientId++, Name = "Помидор", Quantity = 20, Unit = "шт", Cos
    var ingCheese = new Ingredient { Id = nextIngredientId++, Name = "Сыр моцарелла", Quantity = 10, Unit = "кг
    var ingDough = new Ingredient { Id = nextIngredientId++, Name = "Тесто", Quantity = 30, Unit = "кг", CostPe
    var ingChicken = new Ingredient { Id = nextIngredientId++, Name = "Курица", Quantity = 25, Unit = "кг", Cos
    var ingTea = new Ingredient { Id = nextIngredientId++, Name = "Чай листовой", Quantity = 5, Unit = "кг", Co
    var ingWater = new Ingredient { Id = nextIngredientId++, Name = "Вода", Quantity = 200, Unit = "л", CostPer

    ingredients.AddRange(new[] { ingTomato, ingCheese, ingDough, ingChicken, ingTea, ingWater });

    // Блюда
    var pizza = new Dish
    {
        Id = nextDishId++,
        Name = "Пицца Маргарита",
        Category = "Горячее",
        Price = 450
    };
    pizza.Ingredients.Add(new IngredientPortion { Ingredient = ingDough, Amount = 0.3m });
    pizza.Ingredients.Add(new IngredientPortion { Ingredient = ingCheese, Amount = 0.15m });
    pizza.Ingredients.Add(new IngredientPortion { Ingredient = ingTomato, Amount = 2 });

    var salad = new Dish
```

Рисунок 1.10 – Создание класса для блюд

```
var salad = new Dish
{
    Id = nextDishId++,
    Name = "Салат Цезарь",
    Category = "Салаты",
    Price = 390
};
salad.Ingredients.Add(new IngredientPortion { Ingredient = ingChicken, Amount = 0.15m });
salad.Ingredients.Add(new IngredientPortion { Ingredient = ingTomato, Amount = 1 });

var tea = new Dish
{
    Id = nextDishId++,
    Name = "Чай зелёный",
    Category = "Напитки",
    Price = 120
};
tea.Ingredients.Add(new IngredientPortion { Ingredient = ingTea, Amount = 0.01m });
tea.Ingredients.Add(new IngredientPortion { Ingredient = ingWater, Amount = 0.25m });

dishes.AddRange(new[] { pizza, salad, tea });

// Сотрудники
employees.Add(new Employee { Id = nextEmployeeId++, Name = "Иван Петров", Role = EmployeeRole.Chef, Salary = 50000
employees.Add(new Employee { Id = nextEmployeeId++, Name = "Мария Сидорова", Role = EmployeeRole.Waiter, Salary =
employees.Add(new Employee { Id = nextEmployeeId++, Name = "Александр Иванов", Role = EmployeeRole.Waiter, Salary
employees.Add(new Employee { Id = nextEmployeeId++, Name = "Елена Смирнова", Role = EmployeeRole.Manager, Salary =

// Столы
```

Рисунок 1.11 – Создание класса для блюд



```
    // Столы
    tables.Add(new Table { Id = nextTableId++, Seats = 2 });
    tables.Add(new Table { Id = nextTableId++, Seats = 4 });
    tables.Add(new Table { Id = nextTableId++, Seats = 4 });
    tables.Add(new Table { Id = nextTableId++, Seats = 6 });
}
```

Рисунок 1.12 – Создание класса для столов

```
Ссылок: 1
public void Run()
{
    bool running = true;
    while (running)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════════════");
        Console.WriteLine("     ИНФОРМАЦИОННАЯ СИСТЕМА УЧЕТА РЕСТОРАНА");
        Console.WriteLine("═══════════════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Управление меню и ингредиентами");
        Console.WriteLine("2. Управление заказами со столов");
        Console.WriteLine("3. Управление персоналом");
        Console.WriteLine("4. Управление столами и резервированиями");
        Console.WriteLine("5. Отчёты и финансы");
        Console.WriteLine("6. Выход");
        Console.WriteLine();
        Console.Write("Выберите пункт меню (1-6): ");

        string choice = Console.ReadLine();
        Console.WriteLine();
```

Рисунок 1.13 – Создание основного меню приложения

```
string choice = Console.ReadLine();
Console.WriteLine();

switch (choice)
{
    case "1":
        MenuAndIngredientsManagement();
        break;
    case "2":
        OrdersManagement();
        break;
    case "3":
        EmployeesManagement();
        break;
    case "4":
        TablesAndReservationsManagement();
        break;
    case "5":
        ReportsMenu();
        break;
    case "6":
        running = false;
        Console.WriteLine("До свидания!");
        break;
    default:
        Console.WriteLine("Неверный выбор. Нажмите Enter для продолжения...");
        Console.ReadLine();
        break;
}
}
```

Рисунок 1.14 – Реализация выбора

```
Ссылок: 1
private void MenuAndIngredientsManagement()
{
    bool managing = true;
    while (managing)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine("        УПРАВЛЕНИЕ МЕНЮ И ИНГРЕДИЕНТАМИ");
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Просмотреть меню");
        Console.WriteLine("2. Добавить блюдо");
        Console.WriteLine("3. Обновить блюдо");
        Console.WriteLine("4. Удалить блюдо");
        Console.WriteLine("5. Просмотреть ингредиенты");
        Console.WriteLine("6. Добавить ингредиент");
        Console.WriteLine("7. Обновить ингредиент");
        Console.WriteLine("8. Удалить ингредиент");
        Console.WriteLine("9. Вернуться в главное меню");
        Console.WriteLine();
        Console.Write("Выберите действие (1-9): ");
        string choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1": ViewMenu(); break;
            case "2": AddDish(); break;
            case "3": UpdateDish(); break;
            case "4": DeleteDish(); break;
            case "5": ViewIngredients(); break;
            case "6": AddIngredient(); break;
            case "7": UpdateIngredient(); break;
            case "8": DeleteIngredient(); break;
            case "9": managing = false; break;
            default:
                Console.WriteLine("Неверный выбор.");
                Console.ReadLine();
                break;
        }
    }
}
```

Рисунок 1.15 – Создание меню для управления меню и ингредиентами

```
Ссылок: 3
private void ViewMenu()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine("                        МЕНЮ");
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!dishes.Any())
    {
        Console.WriteLine("Меню пусто.");
    }
    else
    {
        var grouped = dishes.GroupBy(d => d.Category);
        foreach (var group in grouped)
        {
            Console.WriteLine($"\nКатегория: {group.Key}");
            Console.WriteLine(new string('-', 60));
            foreach (var dish in group)
            {
                Console.WriteLine(dish);
                if (dish.Ingredients.Any())
                {
                    Console.WriteLine("   Состав:");
                    foreach (var portion in dish.Ingredients)
                    {
                        Console.WriteLine($"     - {portion.Ingredient.Name} {portion.Amount} {portion.Ingredient.Unit}");
                    }
                }
            }
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
```

Рисунок 1.16 – Создание меню для управления меню ресторана

```
Ссылок: 1
private void AddDish()
{
    Console.WriteLine("ДОБАВИТЬ БЛЮДО\n");

    Console.Write("Название блюда: ");
    string name = Console.ReadLine();
    if (string.IsNullOrWhiteSpace(name))
    {
        Console.WriteLine("Название не может быть пустым.");
        Console.ReadLine();
        return;
    }

    Console.Write("Категория: ");
    string category = Console.ReadLine();
    if (string.IsNullOrWhiteSpace(category))
    {
        category = "Без категории";
    }

    Console.Write("Цена: ");
    if (!decimal.TryParse(Console.ReadLine(), NumberStyles.Number, CultureInfo.InvariantCulture, out decimal price) || price < 0)
    {
        Console.WriteLine("Неверная цена.");
        Console.ReadLine();
        return;
    }

    var dish = new Dish
    {
        Id = nextDishId++,
        Name = name,
        Category = category,
        Price = price
    };

    Console.WriteLine("\nДобавить ингредиенты к блюду? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        bool adding = true;
        while (adding)
        {
```

Рисунок 1.17 – Реализация выбора из меню

```
private void DeleteDish()
{
    Console.WriteLine("УДАЛИТЬ БЛЮДО\n");
    ViewMenu();

    Console.Write("\nВведите ID блюда для удаления: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный ID.");
        Console.ReadLine();
        return;
    }

    var dish = dishes.FirstOrDefault(d => d.Id == id);
    if (dish == null)
    {
        Console.WriteLine("Блюдо не найдено.");
        Console.ReadLine();
        return;
    }

    Console.Write($"Удалить '{dish.Name}'? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        dishes.Remove(dish);
        Console.WriteLine("Блюдо удалено.");
    }
    else
    {
        Console.WriteLine("Удаление отменено.");
    }
    Console.ReadLine();
}
```

Рисунок 1.18 – Реализация удаления блюда

```
Ссылок: 3
private void ViewIngredients()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine("                    ИНГРЕДИЕНТЫ");
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!ingredients.Any())
    {
        Console.WriteLine("Ингредиентов нет.");
    }
    else
    {
        foreach (var ing in ingredients.OrderBy(i => i.Name))
        {
            Console.WriteLine(ing);
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}


private void AddIngredient()
{
    Console.WriteLine("ДОБАВИТЬ ИНГРЕДИЕНТ\n");

    Console.Write("Название: ");
    string name = Console.ReadLine();
    if (string.IsNullOrWhiteSpace(name))
    {
        Console.WriteLine("Название не может быть пустым.");
        Console.ReadLine();
        return;
    }

    Console.Write("Единица измерения (кг, л, шт): ");
    string unit = Console.ReadLine();
    if (string.IsNullOrWhiteSpace(unit)) unit = "шт";
```

Рисунок 1.19 – Создание меню для управления ингредиентами

```
Console.Write("\nВведите ID ингредиента: ");
if (!int.TryParse(Console.ReadLine(), out int id))
{
    Console.WriteLine("Неверный ID.");
    Console.ReadLine();
    return;
}

var ing = ingredients.FirstOrDefault(i => i.Id == id);
if (ing == null)
{
    Console.WriteLine("Ингредиент не найден.");
    Console.ReadLine();
    return;
}

Console.WriteLine($"\nТекущие данные: {ing}");

Console.Write("Новое название (Enter - без изменений): ");
string name = Console.ReadLine();
if (!string.IsNullOrWhiteSpace(name)) ing.Name = name;

Console.Write("Новая единица измерения (Enter - без изменений): ");
string unit = Console.ReadLine();
if (!string.IsNullOrWhiteSpace(unit)) ing.Unit = unit;

Console.Write("Новое количество (Enter - без изменений): ");
string qtyInput = Console.ReadLine();
if (!string.IsNullOrWhiteSpace(qtyInput) &&
    decimal.TryParse(qtyInput, NumberStyles.Number, CultureInfo.InvariantCulture, out decimal qty) &&
    qty >= 0)
{
    ing.Quantity = qty;
}

Console.Write("Новая закупочная цена (Enter - без изменений): ");
string costInput = Console.ReadLine();
if (!string.IsNullOrWhiteSpace(costInput) &&
    decimal.TryParse(costInput, NumberStyles.Number, CultureInfo.InvariantCulture, out decimal cost) &&
    cost >= 0)
{
    ing.CostPerUnit = cost;
}
```

Рисунок 1.20 – Реализация работы с ценами

```
private void DeleteIngredient()
{
    Console.WriteLine("УДАЛИТЬ ИНГРЕДИЕНТ\n");
    ViewIngredients();

    Console.Write("\nВведите ID ингредиента: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный ID.");
        Console.ReadLine();
        return;
    }

    var ing = ingredients.FirstOrDefault(i => i.Id == id);
    if (ing == null)
    {
        Console.WriteLine("Ингредиент не найден.");
        Console.ReadLine();
        return;
    }

    Console.Write($"Удалить '{ing.Name}'? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        ingredients.Remove(ing);
        Console.WriteLine("Ингредиент удалён.");
    }
    else
    {
        Console.WriteLine("Удаление отменено.");
    }
    Console.ReadLine();
}
```

Рисунок 1.21 – Реализация удаления ингредиентов

```
// ------ Заказы со столов ------

Ссылок: 1
private void OrdersManagement()
{
    bool managing = true;
    while (managing)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════════");
        Console.WriteLine("            УПРАВЛЕНИЕ ЗАКАЗАМИ");
        Console.WriteLine("═══════════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Создать новый заказ");
        Console.WriteLine("2. Добавить позиции в существующий заказ");
        Console.WriteLine("3. Закрыть заказ и рассчитать счёт");
        Console.WriteLine("4. Просмотреть все заказы");
        Console.WriteLine("5. Вернуться в главное меню");
        Console.WriteLine();
        Console.Write("Выберите действие (1-5): ");
        string choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1": CreateOrder(); break;
            case "2": AddItemsToOrder(); break;
            case "3": CloseOrder(); break;
            case "4": ViewAllOrders(); break;
            case "5": managing = false; break;
            default:
                Console.WriteLine("Неверный выбор.");
                Console.ReadLine();
                break;
        }
    }
}
```

Рисунок 1.22 – Создание меню для управления заказами

```
Ссылок: 1
private void CreateOrder()
{
    Console.WriteLine("СОЗДАНИЕ ЗАКАЗА\n");

    if (!tables.Any())
    {
        Console.WriteLine("Нет столов.");
        Console.ReadLine();
        return;
    }

    Console.WriteLine("Список столов:");
    foreach (var t in tables)
    {
        Console.WriteLine(t);
    }

    Console.Write("Введите номер стола: ");
    if (!int.TryParse(Console.ReadLine(), out int tableId))
    {
        Console.WriteLine("Неверный номер стола.");
        Console.ReadLine();
        return;
    }

    var table = tables.FirstOrDefault(t => t.Id == tableId && t.IsActive);
    if (table == null)
    {
        Console.WriteLine("Стол не найден или не активен.");
        Console.ReadLine();
        return;
    }

    var waiters = employees.Where(e => e.Role == EmployeeRole.Waiter).ToList();
    if (!waiters.Any())
    {
        Console.WriteLine("Нет официантов.");
        Console.ReadLine();
        return;
```

Рисунок 1.23 – Реализация изменения заказов

```
        }

        var waiter = waiters.FirstOrDefault(w => w.Id == waiterId);
        if (waiter == null)
        {
            Console.WriteLine("Официант не найден.");
            Console.ReadLine();
            return;
        }

        var order = new Order
        {
            Id = nextOrderId++,
            Table = table,
            Waiter = waiter,
            OrderTime = DateTime.Now,
            IsClosed = false
        };

        Console.WriteLine("\nДобавление блюд в заказ:");
        AddItemsToOrderInternal(order);

        if (!order.Items.Any())
        {
            Console.WriteLine("Заказ пуст, не будет сохранён.");
        }
        else
        {
            RecalculateOrder(order);
            orders.Add(order);
            Console.WriteLine($"\nЗаказ #{order.Id} создан. Сумма к оплате будет рассчитана при закрытии.");
        }

        Console.ReadLine();
    }
```

Рисунок 1.24 – Работа с официантами

```
Ссылок: 1
private void AddItemsToOrder()
{
    Console.WriteLine("ДОБАВИТЬ ПОЗИЦИИ В ЗАКАЗ\n");
    ViewAllOrders(false);

    Console.Write("\nВведите ID заказа: ");
    if (!int.TryParse(Console.ReadLine(), out int orderId))
    {
        Console.WriteLine("Неверный ID.");
        Console.ReadLine();
        return;
    }

    var order = orders.FirstOrDefault(o => o.Id == orderId);
    if (order == null)
    {
        Console.WriteLine("Заказ не найден.");
        Console.ReadLine();
        return;
    }

    if (order.IsClosed)
    {
        Console.WriteLine("Заказ уже закрыт.");
        Console.ReadLine();
        return;
    }

    AddItemsToOrderInternal(order);
    RecalculateOrder(order);
    Console.WriteLine("\nПозиции добавлены.");
    Console.ReadLine();
}
```

Рисунок 1.25 – Добавление заказов

```
Ссылок: 2
private void AddItemsToOrderInternal(Order order)
{
    bool adding = true;
    while (adding)
    {
        if (!dishes.Any())
        {
            Console.WriteLine("Меню пусто.");
            break;
        }

        Console.WriteLine("\nТекущее содержимое заказа:");
        if (!order.Items.Any())
        {
            Console.WriteLine("  Заказ пуст.");
        }
        else
        {
            foreach (var item in order.Items)
                Console.WriteLine(item);
            Console.WriteLine($"  Сумма без налогов и чаевых: {order.Subtotal,8:C}");
        }

        Console.WriteLine("\nМеню:");
        foreach (var d in dishes)
        {
            Console.WriteLine(d);
        }

        Console.Write("ID блюда (или Enter для завершения): ");
        string input = Console.ReadLine();
        if (string.IsNullOrWhiteSpace(input))
        {
            adding = false;
            break;
        }

        if (!int.TryParse(input, out int dishId))
        {
            Console.WriteLine("Неверный ID.");
            continue;
        }
```

Рисунок 1.26 – Работа с меню ресторана

```
// Проверка и списание ингредиентов со склада

private bool CheckAndConsumeIngredients(Dish dish, int portions)
{
    foreach (var portion in dish.Ingredients)
    {
        decimal need = portion.Amount * portions;
        if (portion.Ingredient.Quantity < need)
            return false;
    }

    foreach (var portion in dish.Ingredients)
    {
        decimal need = portion.Amount * portions;
        portion.Ingredient.Quantity -= need;
    }

    return true;
}


private void RecalculateOrder(Order order)
{
    order.Subtotal = order.Items.Sum(i => i.Subtotal);
    order.Tax = order.Subtotal * TaxRate;
    order.Tips = order.Subtotal * DefaultTipsRate;
    order.Total = order.Subtotal + order.Tax + order.Tips;
}


private void CloseOrder()
{
    Console.WriteLine("ЗАКРЫТИЕ ЗАКАЗА\n");
    ViewAllOrders(false);

    Console.Write("\nВведите ID заказа: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный ID.");
        Console.ReadLine();
        return;
    }
```

Рисунок 1.27 – Проверка и списание ингредиентов

```
    RecalculateOrder(order);

    Console.WriteLine("\nСЧЁТ:");
    foreach (var item in order.Items)
    {
        Console.WriteLine(item);
    }
    Console.WriteLine($"\nСумма:      {order.Subtotal,8:C}");
    Console.WriteLine($"Налог (10%): {order.Tax,8:C}");
    Console.WriteLine($"Чаевые (10%): {order.Tips,8:C}");
    Console.WriteLine($"Итого к оплате: {order.Total,8:C}");

    Console.Write("\nИзменить процент чаевых? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        Console.Write("Новый процент чаевых (0–100): ");
        if (decimal.TryParse(Console.ReadLine(), NumberStyles.Number, CultureInfo.InvariantCulture, out decimal tipsPercent) && tips
        {
            order.Tips = order.Subtotal * (tipsPercent / 100m);
            order.Total = order.Subtotal + order.Tax + order.Tips;
            Console.WriteLine($"Новая сумма чаевых: {order.Tips,8:C}");
            Console.WriteLine($"Итого к оплате: {order.Total,8:C}");
        }
    }

    Console.Write("\nПодтвердить закрытие заказа? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        order.IsClosed = true;
        Console.WriteLine("Заказ закрыт.");
```

Рисунок 1.28 – Расчёт чаевых

```
private void ViewAllOrders(bool waitKey = true)
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════════════");
    Console.WriteLine("                          ЗАКАЗЫ");
    Console.WriteLine("═══════════════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!orders.Any())
    {
        Console.WriteLine("Заказов нет.");
    }
    else
    {
        foreach (var o in orders.OrderByDescending(o => o.OrderTime))
        {
            Console.WriteLine(o);
        }
    }

    if (waitKey)
    {
        Console.WriteLine("\nНажмите Enter для продолжения...");
        Console.ReadLine();
    }
}
```

Рисунок 1.29 – Меню для работы с заказами

```
Ссылок: 1
private void EmployeesManagement()
{
    bool managing = true;
    while (managing)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine("              УПРАВЛЕНИЕ ПЕРСОНАЛОМ");
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Просмотреть сотрудников");
        Console.WriteLine("2. Добавить сотрудника");
        Console.WriteLine("3. Обновить сотрудника");
        Console.WriteLine("4. Удалить сотрудника");
        Console.WriteLine("5. Вернуться в главное меню");
        Console.WriteLine();
        Console.Write("Выберите действие (1-5): ");
        string choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1": ViewEmployees(); break;
            case "2": AddEmployee(); break;
            case "3": UpdateEmployee(); break;
            case "4": DeleteEmployee(); break;
            case "5": managing = false; break;
            default:
                Console.WriteLine("Неверный выбор.");
                Console.ReadLine();
                break;
        }
    }
}
```

Рисунок 1.30 – Меню для управления персоналом

```
Ссылок: 3
private void ViewEmployees()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine("                    СОТРУДНИКИ");
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!employees.Any())
    {
        Console.WriteLine("Сотрудников нет.");
    }
    else
    {
        var grouped = employees.GroupBy(e => e.Role);
        foreach (var group in grouped)
        {
            Console.WriteLine($"\n{group.Key}:");
            foreach (var e in group)
                Console.WriteLine(e);
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}

Ссылок: 1
private void AddEmployee()
{
    Console.WriteLine("ДОБАВИТЬ СОТРУДНИКА\n");

    Console.Write("ФИО: ");
    string name = Console.ReadLine();
    if (string.IsNullOrWhiteSpace(name))
    {
        Console.WriteLine("Имя не может быть пустым.");
        Console.ReadLine();
        return;
    }
```

Рисунок 1.31 – Меню для управления сотрудниками

```
private void DeleteEmployee()
{
    Console.WriteLine("УДАЛИТЬ СОТРУДНИКА\n");
    ViewEmployees();

    Console.Write("\nВведите ID сотрудника: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный ID.");
        Console.ReadLine();
        return;
    }

    var emp = employees.FirstOrDefault(e => e.Id == id);
    if (emp == null)
    {
        Console.WriteLine("Сотрудник не найден.");
        Console.ReadLine();
        return;
    }
```

Рисунок 1.32 – Меню для управления сотрудниками

```
private void TablesAndReservationsManagement()
{
    bool managing = true;
    while (managing)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════════");
        Console.WriteLine("     УПРАВЛЕНИЕ СТОЛАМИ И РЕЗЕРВИРОВАНИЕМ");
        Console.WriteLine("═══════════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Просмотреть столы");
        Console.WriteLine("2. Добавить стол");
        Console.WriteLine("3. Обновить стол");
        Console.WriteLine("4. Удалить стол");
        Console.WriteLine("5. Просмотреть резервирования");
        Console.WriteLine("6. Создать резервирование");
        Console.WriteLine("7. Отменить резервирование");
        Console.WriteLine("8. Вернуться в главное меню");
        Console.WriteLine();
        Console.Write("Выберите действие (1-8): ");
        string choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1": ViewTables(); break;
            case "2": AddTable(); break;
            case "3": UpdateTable(); break;
            case "4": DeleteTable(); break;
            case "5": ViewReservations(); break;
            case "6": CreateReservation(); break;
            case "7": CancelReservation(); break;
            case "8": managing = false; break;
            default:
                Console.WriteLine("Неверный выбор.");
                Console.ReadLine();
                break;
        }
    }
}
```

Рисунок 1.32 – Меню для управления столами и резервирования

```
private void ViewTables()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════════");
    Console.WriteLine("                          СТОЛЫ");
    Console.WriteLine("═══════════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!tables.Any())
    {
        Console.WriteLine("Столов нет.");
    }
    else
    {
        foreach (var t in tables.OrderBy(t => t.Id))
        {
            Console.WriteLine(t);
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
```

Рисунок 1.33 – Меню для работы столами

```
private void UpdateTable()
{
    Console.WriteLine("ОБНОВИТЬ СТОЛ\n");
    ViewTables();

    Console.Write("\nВведите номер стола: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный номер.");
        Console.ReadLine();
        return;
    }

    var table = tables.FirstOrDefault(t => t.Id == id);
    if (table == null)
    {
        Console.WriteLine("Стол не найден.");
        Console.ReadLine();
        return;
    }

    Console.Write("Новое количество мест (Enter – без изменений): ");
    string seatsInput = Console.ReadLine();
    if (!string.IsNullOrWhiteSpace(seatsInput) &&
        int.TryParse(seatsInput, out int seats) &&
        seats > 0)
    {
        table.Seats = seats;
    }
```

Рисунок 1.34 – Реализация управления столов

```
private void DeleteTable()
{
    Console.WriteLine("УДАЛИТЬ СТОЛ\n");
    ViewTables();

    Console.Write("\nВведите номер стола: ");
    if (!int.TryParse(Console.ReadLine(), out int id))
    {
        Console.WriteLine("Неверный номер.");
        Console.ReadLine();
        return;
    }

    var table = tables.FirstOrDefault(t => t.Id == id);
    if (table == null)
    {
        Console.WriteLine("Стол не найден.");
        Console.ReadLine();
        return;
    }

    Console.Write($"Удалить стол #{table.Id}? (д/н): ");
    if (Console.ReadLine().Trim().ToLower() == "д")
    {
        tables.Remove(table);
        Console.WriteLine("Стол удалён.");
    }
    else
    {
        Console.WriteLine("Удаление отменено.");
    }

    Console.ReadLine();
}
```

Рисунок 1.35 – Реализация удаления столов

```
Ссылок: 2
private void ViewReservations()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════════════");
    Console.WriteLine("                    РЕЗЕРВИРОВАНИЯ");
    Console.WriteLine("═══════════════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!reservations.Any())
    {
        Console.WriteLine("Резервирований нет.");
    }
    else
    {
        foreach (var r in reservations.OrderBy(r => r.ReservationTime))
        {
            Console.WriteLine(r);
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
```

Рисунок 1.35 – Меню для резервирования столов

```
    // простая проверка пересечения по времени (±2 часа)
    var conflict = reservations.Any(r =>
        r.Table.Id == table.Id &&
        Math.Abs((r.ReservationTime - time).TotalMinutes) < 120);

    if (conflict)
    {
        Console.WriteLine("На это время стол уже зарезервирован.");
        Console.ReadLine();
        return;
    }

    reservations.Add(new Reservation
    {
        Id = nextReservationId++,
        Table = table,
        CustomerName = name,
        Phone = phone,
        GuestsCount = guests,
        ReservationTime = time
    });

    Console.WriteLine("\nРезервирование создано.");
    Console.ReadLine();
}
```

Рисунок 1.36 – Простая проверка пересечения по времени

```
Ссылок: 1
private void ReportsMenu()
{
    bool managing = true;
    while (managing)
    {
        Console.Clear();
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine("                    ОТЧЁТЫ И ФИНАНСЫ");
        Console.WriteLine("═══════════════════════════════════════════════════════");
        Console.WriteLine();
        Console.WriteLine("1. Ежедневный отчёт выручки");
        Console.WriteLine("2. Отчёт по расходам на продукты");
        Console.WriteLine("3. Краткая сводка по персоналу");
        Console.WriteLine("4. Вернуться в главное меню");
        Console.WriteLine();
        Console.Write("Выберите действие (1-4): ");
        string choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1": DailyRevenueReport(); break;
            case "2": IngredientsCostReport(); break;
            case "3": StaffSummaryReport(); break;
            case "4": managing = false; break;
            default:
                Console.WriteLine("Неверный выбор.");
                Console.ReadLine();
                break;
        }
    }
}
```

Рисунок 1.37 – Меню для управления отчетами и финансами

```
Ссылок: 1
private void DailyRevenueReport()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════");
    Console.WriteLine("                ЕЖЕДНЕВНЫЙ ОТЧЁТ ВЫРУЧКИ");
    Console.WriteLine("═══════════════════════════════════════════════════");
    Console.WriteLine();

    Console.Write("Введите дату (дд.ММ.гггг) или Enter для сегодняшней: ");
    string input = Console.ReadLine();
    DateTime date;
    if (string.IsNullOrWhiteSpace(input))
        date = DateTime.Today;
    else if (!DateTime.TryParseExact(input, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date))
    {
        Console.WriteLine("Неверный формат даты.");
        Console.ReadLine();
        return;
    }

    var dailyOrders = orders.Where(o => o.OrderTime.Date == date.Date && o.IsClosed).ToList();

    if (!dailyOrders.Any())
    {
        Console.WriteLine("Закрытых заказов за выбранный день нет.");
    }
    else
    {
        decimal totalRevenue = dailyOrders.Sum(o => o.Total);
        decimal totalTax = dailyOrders.Sum(o => o.Tax);
        decimal totalTips = dailyOrders.Sum(o => o.Tips);
```

Рисунок 1.38 – Меню для управления отчетами выручки

```
        Console.WriteLine($"Дата: {date:dd.MM.yyyy}");
        Console.WriteLine($"Всего заказов: {dailyOrders.Count}");
        Console.WriteLine($"Выручка (с налогами и чаевыми): {totalRevenue,8:C}");
        Console.WriteLine($"Из них налоги: {totalTax,8:C}");
        Console.WriteLine($"Чаевые: {totalTips,8:C}");
        Console.WriteLine();

        var byWaiter = dailyOrders.GroupBy(o => o.Waiter.Name)
            .Select(g => new
            {
                Waiter = g.Key,
                Orders = g.Count(),
                Sum = g.Sum(o => o.Total)
            });

        Console.WriteLine("Выручка по официантам:");
        foreach (var w in byWaiter)
        {
            Console.WriteLine($"  {w.Waiter,-20} Заказов: {w.Orders,3} | Сумма: {w.Sum,8:C}");
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
```

Рисунок 1.39 – Реализация расчёта выручки официантов

```
private void IngredientsCostReport()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine("           ОТЧЁТ ПО РАСХОДАМ НА ПРОДУКТЫ");
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!ingredients.Any())
    {
        Console.WriteLine("Ингредиентов нет.");
    }
    else
    {
        decimal totalStockCost = ingredients.Sum(i => i.Quantity * i.CostPerUnit);
        Console.WriteLine($"Текущая стоимость складских запасов: {totalStockCost,8:C}");
        Console.WriteLine();
        Console.WriteLine("Ингредиенты:");
        foreach (var i in ingredients)
        {
            decimal cost = i.Quantity * i.CostPerUnit;
            Console.WriteLine($"  {i.Name,-20} Остаток: {i.Quantity,8:F2} {i.Unit,-4} | Стоимость: {cost,8:C}");
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
```

Рисунок 1.40 – Меню отчета по расходам на продукты

```
Ссылок: 1
private void StaffSummaryReport()
{
    Console.Clear();
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine("              КРАТКАЯ СВОДКА ПО ПЕРСОНАЛУ");
    Console.WriteLine("═══════════════════════════════════════════════════════");
    Console.WriteLine();

    if (!employees.Any())
    {
        Console.WriteLine("Сотрудников нет.");
    }
    else
    {
        decimal totalSalary = employees.Sum(e => e.Salary);
        Console.WriteLine($"Всего сотрудников: {employees.Count}");
        Console.WriteLine($"Общий фонд зарплаты: {totalSalary,8:C}");
        Console.WriteLine();

        var grouped = employees.GroupBy(e => e.Role);
        foreach (var g in grouped)
        {
            Console.WriteLine($"{g.Key}: {g.Count()} чел., суммарная зарплата: {g.Sum(e => e.Salary),8:C}");
        }
    }

    Console.WriteLine("\nНажмите Enter для продолжения...");
    Console.ReadLine();
}
}
```

Рисунок 1.41 – Меню отчета краткой сводки по персоналу

```
Ссылок: 0
internal class Program
{
    Ссылок: 0
    static void Main(string[] args)
    {
        var system = new RestaurantSystem();
        system.Run();
    }
}
```

Рисунок 1.42 – Запуск программы

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА

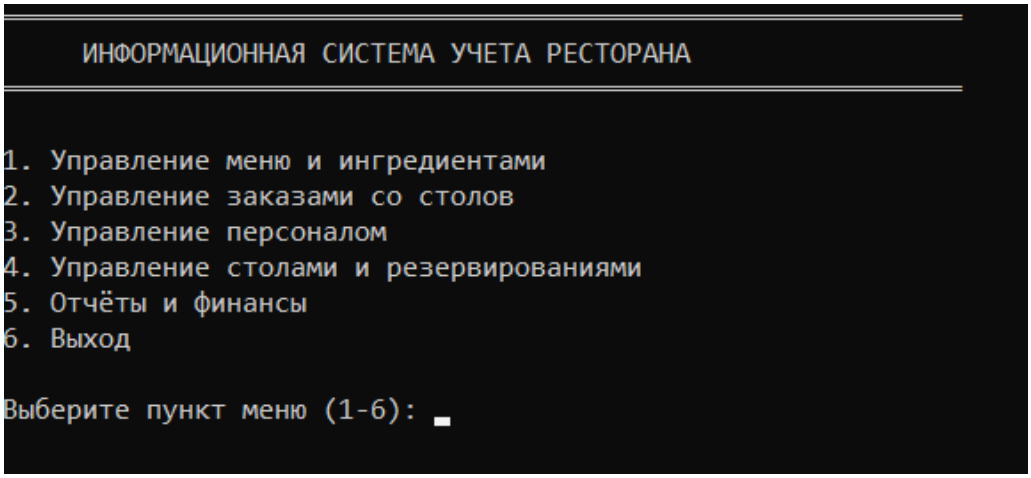


Рисунок 2.1– Запуск программы

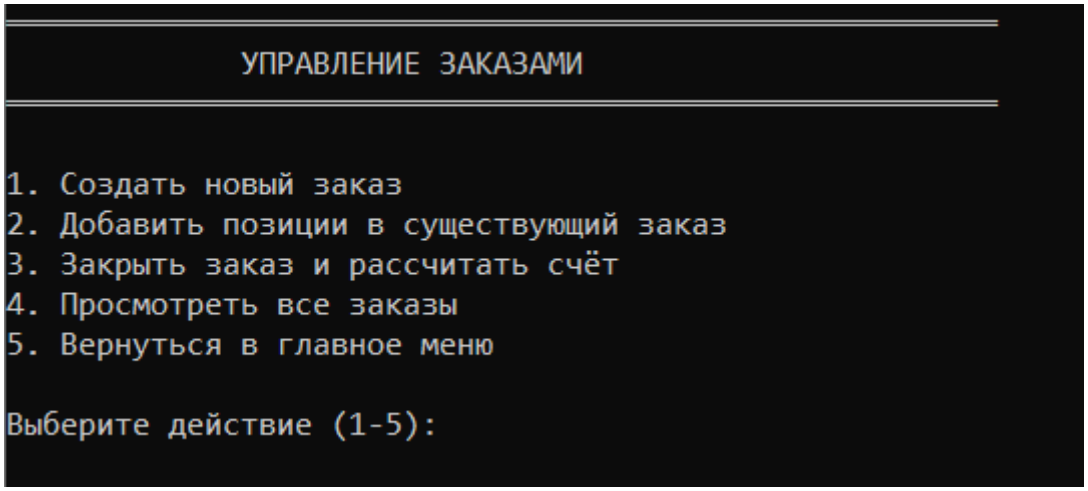


Рисунок 2.2– Управление заказами

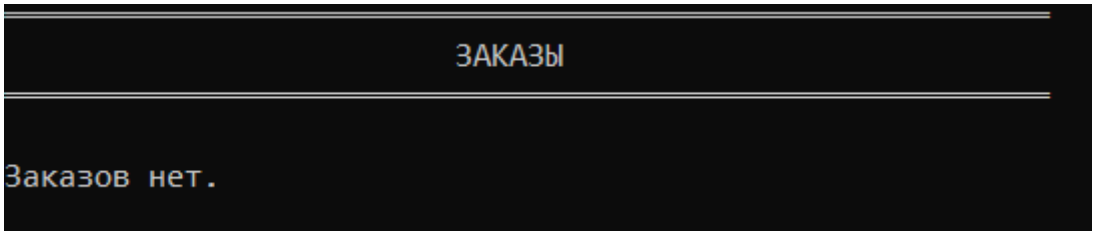


Рисунок 2.3– Вывод информации об отсутствии заказов

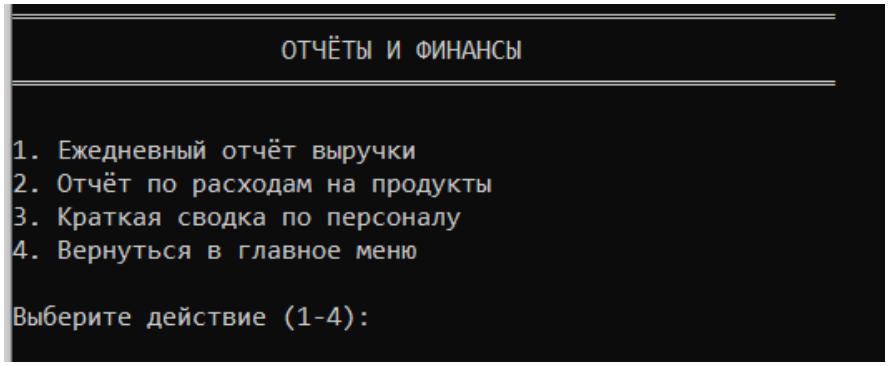


Рисунок 2.4– Вывод информации об отчетах и финансах

ЗАКЛЮЧЕНИЕ

В ходе выполнения учебной практики было успешно разработано полнофункциональное консольное приложение для управления ресторанов, которое продемонстрировало глубокое понимание принципов объектно-ориентированного программирования и практические навыки в разработке на платформе .NET. Приложение включает комплексную функциональность: управление меню и ингредиентами с отслеживанием цен, системы управления заказами и персоналом, организацию резервирования столов с проверкой пересечений по времени, а также развитую систему отчетности, включающую финансовые показатели и статистику по расходам. В процессе реализации был получен практический опыт создания пользовательских интерфейсов, разработки логики бизнес-процессов и проектирования архитектуры приложения. Полученные навыки являются основой для дальнейшего профессионального развития в области разработки программного обеспечения, а сам проект может служить примером применения теоретических знаний в реальной практической задаче. Успешное завершение данного проекта подтверждает готовность к разработке более сложных систем и демонстрирует способность к самостоятельному решению технических задач в рамках разработки приложений.