|  |
| --- |
| МИНОБРНАУКИ РОССИИ  ОРСКИЙ ГУМАНИТАРНО-ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)  ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ  «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  (ОРСКИЙ ГУМАНИТАРНО-ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ  (ФИЛИАЛ) ОГУ)  Факультет инженерии, экономики и права  Кафедра программного обеспечения |
| **ОТЧЕТ**  по практике  *(практика по получению первичных профессиональных умений и навыков*  *научно-исследовательской деятельности)*  Орский гуманитарно-технологический институт (филиал) ОГУ  7024. 783П     |  |  |  | | --- | --- | --- | | Руководитель от кафедры  Старший преподаватель | подпись, дата | А. В. Парамонов | |  |  |  | | Исполнитель  студент группы | подпись, дата | Е. Ю. Инжуватова |   Орск 2024 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Утверждаю  Заведующий кафедрой программного обеспечения  (наименование кафедры) | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | А. С. Попов | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_\_г. | | |

**ЗАДАНИЕ**

**на выполнение отчета по учебной практике**

*(практика по получению первичных профессиональных умений и навыков научно-исследовательской деятельности)*

студенту \_\_\_\_\_\_\_\_\_\_Инжуватовой Екатерине Юрьевне\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

по направлению подготовки (специальности) 09.03.01 ИВТ профиль ПОСВТАС\_\_\_\_\_

по учебной практике *(практика по получению первичных профессиональных умений и навыков научно-исследовательской деятельности)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

наименование дисциплины, модуля

1 Учебная практика (работа): \_Создание информационной системы по учёту заказов в ресторане\_\_

2 Срок сдачи студентом отчета (работы) «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.

3 Цель и задачи отчета (работы) разработка системы учета заказов, предоставление пользователю необходимой информации

4 Исходные данные к отчету (работе) \_создать информационную систему по учёту заказов в заведении, и предоставить пользователю информацию о заказах

5 Перечень вопросов, подлежащих разработке: актуальность, автоматизация, язык программирования C#, ключевые компоненты системы, создание проекта, классы, класс для хранения данных, исполняемый файл, тестирование программы

6 Перечень графического (иллюстрированного) материала: 9 изображений пользовательского интерфейса

Дата выдачи и получения задания: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г

Руководитель «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_А.С.Попов\_\_

(подпись) (инициалы, фамилия)

Студент «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Ю.Инжуватова

(подпись) (инициалы, фамилия)

**Аннотация**

Отчет по учебной практике содержит 28 страниц, в том числе 9 рисунков, 8 источников.

В работе исследован процесс разработки информационной системы для учета заказов в ресторане. Основное внимание уделено проектированию и реализации различных модулей, таких как интерфейс для ввода и обработки заказов, управление продуктами и категориями, а также просмотр и анализ созданных заказов.

После внедрения системы проведены тестирования и анализы для изучения ее эффективности и удобства использования, включая оптимизацию алгоритмов обработки данных и создание интуитивного интерфейса для пользователей.

# 

**Дневник**

Дневник прохождения Учебной практики (научно–исследовательская работа (получение первичных навыков научно–исследовательской работы))

|  |  |  |  |
| --- | --- | --- | --- |
| Вид выполняемой работы | Календарные  сроки | Рабочее место студента | Ответственный специалист  кафедры |
| Вводная лекция.  Получение  задания | 01.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Сбор материала для выполнения учебной практики | 01.07.2024 – 02.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Изучение  теоретического материала  учебной практики | 01.07.2024 – 03.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Составление практической  части работы | 04.07.2024 – 08.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Тестирование разрабатываемого продукта для учебной практики | 09.07.2024 – 10.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Составление и оформление  отчета | 4.07.2024 – 11.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Редактирование отчета | 12.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |
| Защита отчета по учебной практике | 13.07.2024 | Компьютерный класс 4–117 | Парамонов А.В. |

Подпись руководителя практики от кафедры \_\_\_\_\_\_\_ Парамонов А. В.

**Содержание**

|  |  |
| --- | --- |
| Введение……………………………………………………………………………… | 6 |
| 1. Теоретическая часть………………………………………………………………. | 7 |
| 1.1 Актуальность………...…………………………………………………………... | 7 |
| 1.2 Автоматизация....……………………………………………….……................... | 7 |
| 1.3 Язык программирования C#…………………………………………………..… | 8 |
| 1.4 Ключевые компоненты системы………………………………………………... | 8 |
| 2. Практическая часть………………………………………..……………………… | 9 |
| 2.1 Создание проекта...……………………………………………………………… | 9 |
| 2.2 Классы…………………………………...……………………………………….. | 9 |
| 2.3 Класс для хранения данных……………………………………………………... | 12 |
| 2.4 Исполняемый файл…….………………………………………………………… | 16 |
| 2.5 Тестирование программы……………………………………………………….. | 24 |
| Заключение…………………………………………………………………………… | 27 |
| Список литературы…………………………………………………………………... | 28 |

# Введение

# Информационные технологии в общественном питании – это отображенное в проектном формате концентрированное представление научных знаний, сведений и практического опыта, которое позволяет рациональным образом организовать информационные процессы в сфере общественного питания.

Улучшение производительности различных предприятий за счет «Информационных технологий» осуществляется путем оснащения различных структурных подразделений с различными профессиональными программными продуктами.

Использование современных информационных технологий в сфере управления обеспечивает повышение качества экономической информации, ее точности, объективности и как следствие этого, возможности принятия своевременных управленческих решений.

Очевидны неоспоримые преимущества автоматизированного ресторана перед другими подобными заведениями: высокое качество сервиса и скорость обслуживания клиентов, отсутствие ошибок при оформлении заказа, обработка и передача заказа в автоматическом режиме, абсолютный контроль всех процессов от момента приема заказа до его исполнения, возможность непрерывно отслеживать финансовые результаты работы заведения.

В данном отчете для учебной практики будет рассмотрено описание и выбор алгоритма обработки текстовых запросов, проектирование и реализация системы, а также эффективность его работы и возможности для дальнейшего улучшения.

# 1 Теоретическая часть

## Актуальность

Информационные технологии берут на себя процедуру формирования блюда на основе набора ингредиентов (произвольно и часто меняющегося во времени) и схемы закупки продуктов (не только от организаций, но и с рынка по закупочным актам).

Автоматизируется ведение списка блюд с учетом нормативов расхода продуктов, сезонных норм закладок продуктов в блюда, «вложенности блюд» на основе использования справочников продуктов и блюд (список ингредиентов может включать в себя как продукты, так и другие блюда с любой степенью вложенности).

Современные приложения автоматически определяют расход ингредиентов по каждому блюду, списывают нужное количество продуктов и рассчитывают себестоимость блюд, формируют калькуляционные карточки на блюда в условиях динамического изменения закупочных, учетных и продажных цен.

В большинстве информационных систем заложены возможности ведение количественно-суммового учета продуктов и блюд с контролируемым процентом наценки при назначении розничных цен на блюдо, а также ведение «истории» отпускных цен на блюда и продукты.

Информационные технологии позволяют значительно упростить и создают более строгое ведение учета продуктов и блюд на нескольких кухнях и точках реализации. В данном случае информационные системы позволяют автоматически устанавливать разные цены на блюда и услуги, в зависимости от места реализации и единиц измерения. За счет автоматизации упрощается процесс движения товаров, например, организация поступления товаров непосредственно на кухню или место реализации, минуя кладовую или перемещение продуктов на кухни для производства блюда, либо в розничную продажу на точки реализации.

Говоря иначе современные информационные технологии, которые разработаны специально для сферы общественного питания, позволяют значительно упростить, оптимизировать и ускорить целый ряд рутинных повседневных, специфических для этого бизнеса операций

## Автоматизация

Автоматизация в общественном питании – это осознанный и просто необходимый шаг для предприятия, которое, в перспективе, хочет перейти на более качественный уровень работы. Уровень, где снижается влияние человеческого фактора при обслуживании клиентов и где анализ показателей позволяет вести плановую деятельность, правильно распределять ресурсы и средства фирмы. Все это, в итоге, приводит к повышению доходов предприятия общественного питания.

Таким образом, можно выделить следующие положительные стороны автоматизации организаций в сфере общественного питания:

1. ускоряется работа персонала;
2. официанту проще принимать заказы от клиентов;
3. бухгалтеру или менеджеру проще следить за складскими остатками, сводить отчеты, составлять калькуляционные карты;
4. возможность провести проверку в любой момент снижают различного рода махинации персонала.

## Язык программирования C#

С# – это объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

С# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

Популярность языка – еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft. Программисты, хорошо знакомые с С#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество.

Понятный синтаксис C# заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок при работе над приложениями в больших командах.

# Ключевые компоненты системы

Каждая система учета заказов для ресторана имеет ключевые компоненты:

1. Продукты ресторана: Информация о блюдах, включающая в себя название, описание блюд, цену, питательные свойства.
2. Категории продуктов ресторана.
3. Заказы: Информация, которая включает в себя номер и позицию заказа, а также место (в зале или с собой) и дату создания.
4. Позиция заказов: Информация, которая включает в себя идентификатор, продукт и его цену, а также количество.

# Практическая часть

# Создание проекта

Создаем внутри проекта папку Domain. Внутри папки Domain создаем папку Entities, внутрь которой размещаем классы Product.cs, Category.cs, Order.cs, OrderItem.cs. Также создаем вспомогательный класс ApplicationContext.cs в папке Domain для хранения данных. Основной программный файл Program.cs находится вне папки Domain. На рисунке 2.1 показано дерево всех файлов.

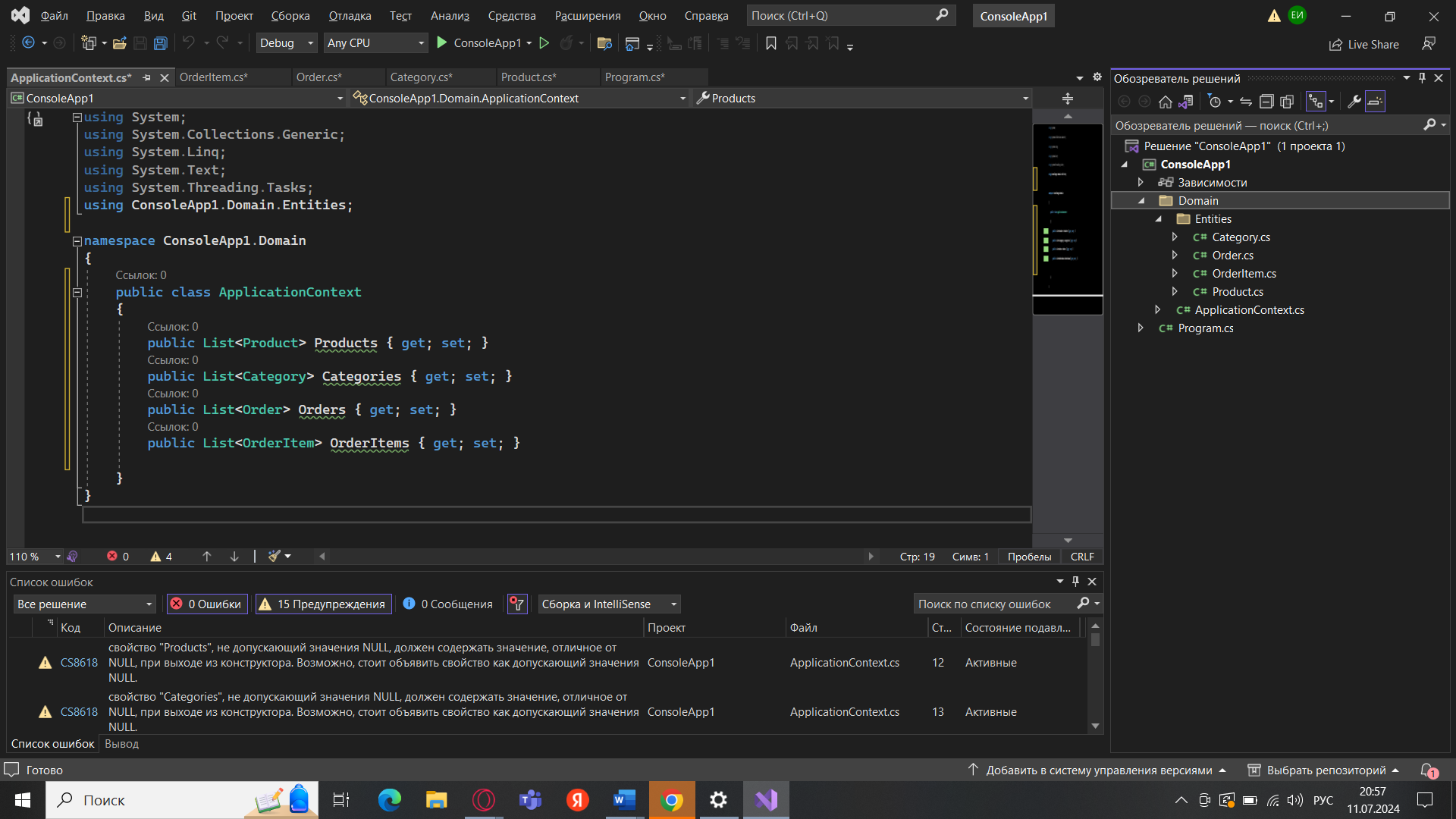


Рисунок 2.1 – Дерево файлов

**2.2 Классы**

Публичный класс Product описывает продукт ресторана. Класс содержит следующие данные: идентификатор, название, описание, категории (список объектов класса Category, см. класс List), масса белка (в граммах на 100 граммов продукта), масса жиров (в граммах на 100 граммов продукта), масса углеводов (в граммах на 100 граммов продукта), количество калорий (в килокалориях), цена. Идентификатор должен быть открыт для чтения, но закрыт для записи. Остальные данные – открыты для чтения и записи. Создать конструктор с одним параметром – идентификатором. На рисунке 2.2 показан код данного класса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2.2 – Код класса Product.cs

Публичный класс Category описывает категорию продукта ресторана (например, пицца или десерты). Класс содержит следующие данные: идентификатор, название. Идентификатор должен быть открыт для чтения, но закрыт для записи. Остальные данные – открыты для чтения и записи. Создать конструктор с одним параметром – идентификатором. На рисунке 2.3 представлен код данного класса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2.3 – Код класса Category.cs

Публичный класс Order описывает заказ. Класс содержит следующие данные: идентификатор, номер заказа (например, A-51 или A-54, где буква A постоянна, а номер увеличивается от 00 до 99 и сбрасывается в 00 при достижении 99), позиции заказа (список объектов класса OrderItem, см. класс List), место (в зале или с собой), дата создания заказа. Идентификатор должен быть открыт для чтения, но закрыт для записи. Остальные данные – открыты для чтения и записи. Создать конструктор с одним параметром – идентификатором. На рисунке 2.4 продемонстрирован код данного класса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2.4 – Код класса Order.cs

Публичный класс OrderItem описывает позицию заказа (например, чизкейк x 3). Класс содержит следующие данные: идентификатор, заказ (объект класса Order), продукт (объект класса Product), количество, цена продукта. Идентификатор должен быть открыт для чтения, но закрыт для записи. Остальные данные – открыты для чтения и записи. Создать конструктор с одним параметром – идентификатором. На рисунке 2.5 показан код данного класса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2.5 – Код класса OrderItem.cs

## 2.3 Класс для хранения данных

Класс ApplicationContext нужен для хранения данных и создается в папке Domain. Он описывает взаимодействие списков с соответствующими переменными.

Ниже представлен код данного класса.

using ConsoleApp1.Domain.Entities;

using System.Collections.Generic;

namespace ConsoleApp1.Domain

{

public class ApplicationContext

{

public List<Category> Categories { get; set; } = new List<Category>();

public List<Product> Products { get; set; } = new List<Product>();

public List<Order> Orders { get; set; } = new List<Order>();

public List<OrderItem> OrderItems { get; set; } = new List<OrderItem>();

public ApplicationContext()

{

// Предварительное заполнение данных

SeedData();

}

private void SeedData()

{

var categories = new[]

{

new Category(0) { Name = "Пицца" },

new Category(1) { Name = "Бургеры" },

new Category(2) { Name = "Напитки" },

new Category(3) { Name = "Гарниры" },

new Category(4) { Name = "Десерты" }

};

// Создание продуктов

var products = new[]

{

new Product(0)

{

Name = "Пепперони",

Description = "Классическая пицца с пикантной пепперони, моцареллой и томатным соусом.",

Categories = new List<Category> { categories[0] },

Protein = 15.0,

Fats = 11.2,

Carbohydrates = 20.6,

Calories = 269.3,

Price = 285.0m

},

new Product(1)

{

Name = "Маргарита",

Description = "Простая, но элегантная пицца с томатным соусом, моцареллой и свежим базиликом.",

Categories = new List<Category> { categories[0] },

Protein = 10.5,

Fats = 9.0,

Carbohydrates = 24.0,

Calories = 208.0,

Price = 260.0m

},

new Product(2)

{

Name = "Чизбургер",

Description = "Сочный говяжий бургер с расплавленным сыром чеддер, луком, помидорами и маринованными огурцами.",

Categories = new List<Category> { categories[1] },

Protein = 16.0,

Fats = 12.0,

Carbohydrates = 31.0,

Calories = 302.0,

Price = 100.0m

},

new Product(3)

{

Name = "Воппер",

Description = "Большой, сочный бургер с говяжьей котлетой, салатом, помидором, луком и майонезом.",

Categories = new List<Category> { categories[1] },

Protein = 21.3,

Fats = 33.0,

Carbohydrates = 50.0,

Calories = 289.0,

Price = 120.0m

},

new Product(4)

{

Name = "Кола",

Description = "Классический газированный напиток, освежающий и утоляющий жажду.",

Categories = new List<Category> { categories[2] },

Protein = 0.0,

Fats = 0.0,

Carbohydrates = 10.6,

Calories = 42.0,

Price = 50.0m

},

new Product(5)

{

Name = "Фанта",

Description = "Свежий и фруктовый газированный напиток с ароматом цитрусовых.",

Categories = new List<Category> { categories[2] },

Protein = 0.0,

Fats = 0.0,

Carbohydrates = 8.0,

Calories = 33.0,

Price = 50.0m

},

new Product(6)

{

Name = "Картофель фри",

Description = "Золотистые, хрустящие картофельные брусочки, идеально сочетающиеся с любым блюдом.",

Categories = new List<Category> { categories[3] },

Protein = 3.8,

Fats = 15.5,

Carbohydrates = 30.0,

Calories = 276.0,

Price = 170.0m

},

new Product(7)

{

Name = "Кукурузные шарики",

Description = "Нежные кукурузные шарики, обжаренные до золотистой корочки, с легким сливочным вкусом.",

Categories = new List<Category> { categories[3] },

Protein = 7.0,

Fats = 37.0,

Carbohydrates = 50.0,

Calories = 350.0,

Price = 130.0m

},

new Product(8)

{

Name = "Чизкейк",

Description = "Нежный чизкейк с нежной текстурой и классическим сливочным вкусом.",

Categories = new List<Category> { categories[4] },

Protein = 3.0,

Fats = 21.0,

Carbohydrates = 22.0,

Calories = 293.0,

Price = 120.0m

},

new Product(9)

{

Name = "Сандей",

Description = "Холодный и освежающий десерт из мороженого, взбитых сливок и фруктового соуса.",

Categories = new List<Category> { categories[4] },

Protein = 4.6,

Fats = 5.0,

Carbohydrates = 36.0,

Calories = 223.0,

Price = 110.0m

}

};

Products.AddRange(products);

Categories.AddRange(categories);

}

}

}

# 2.4 Исполняемый файл

Исполняемый файл Program.cs берет данные из ApplicationContext и подключается через конструкторы в остальные файлы в папке Entities.

Ниже представлен его код.

using ConsoleApp1.Domain;

using ConsoleApp1.Domain.Entities;

using ConsoleTables;

using System;

using System.Linq;

namespace ConsoleApp1

{

public class Program

{

static ApplicationContext context = new ApplicationContext();

static Order currentOrder = null;

static void Main(string[] args)

{

MainMenu();

}

static void MainMenu()

{

while (true)

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Введите номер действия.");

Console.WriteLine("1. В зале");

Console.WriteLine("2. С собой");

Console.WriteLine("3. Просмотр заказов");

Console.WriteLine("[q – завершить работу]");

string input = Console.ReadLine();

if (input == "q")

{

break;

}

switch (input)

{

case "1":

StartOrder("в зале");

break;

case "2":

StartOrder("с собой");

break;

case "3":

ViewOrders();

break;

default:

Console.WriteLine("Неверный ввод, попробуйте еще раз.");

break;

}

}

}

static void StartOrder(string place)

{

currentOrder = new Order(context.Orders.Count + 1)

{

Place = place,

CreationDate = DateTime.Now,

OrderNumber = GenerateOrderNumber()

};

SelectCategory();

}

static void SelectCategory()

{

while (true)

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Введите категорию.");

for (int i = 0; i < context.Categories.Count; i++)

{

Console.WriteLine($"{i + 1}. {context.Categories[i].Name}");

}

Console.WriteLine("[b – назад, p – оформить заказ, c - отменить заказ, q – завершить работу]");

string input = Console.ReadLine();

if (input == "b")

{

return;

}

else if (input == "p")

{

ConfirmOrder();

return;

}

else if (input == "c")

{

CancelOrder();

return;

}

else if (input == "q")

{

Environment.Exit(0);

}

if (int.TryParse(input, out int categoryIndex) && categoryIndex > 0 && categoryIndex <= context.Categories.Count)

{

SelectProduct(context.Categories[categoryIndex - 1]);

}

else

{

Console.WriteLine("Неверный ввод, попробуйте еще раз.");

}

}

}

static void SelectProduct(Category category)

{

while (true)

{

Console.Clear();

Console.WriteLine($"Терминал для заказа. Категория: {category.Name}");

Console.WriteLine("Выберите продукт:");

var products = context.Products

.Where(p => p.Categories.Any(c => c.Id == category.Id))

.ToList();

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

{

Console.WriteLine($"{i + 1}. {products[i].Name} ({products[i].Price} руб.)");

}

Console.WriteLine("[b - назад, p - оформить заказ, c - отменить заказ, q - завершить работу]");

string input = Console.ReadLine();

if (input == "b")

{

return;

}

else if (input == "p")

{

ConfirmOrder();

return;

}

else if (input == "c")

{

CancelOrder();

return;

}

else if (input == "q")

{

Environment.Exit(0);

}

if (int.TryParse(input, out int productIndex) && productIndex > 0 && productIndex <= products.Count)

{

AddProductToOrder(products[productIndex - 1]);

}

else

{

Console.WriteLine("Неверный ввод, попробуйте еще раз.");

}

}

}

static void AddProductToOrder(Product product)

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine($"Введите количество для {product.Name}:");

Console.WriteLine("[b – назад, p – оформить заказ, c - отменить заказ, q – завершить работу]");

string input = Console.ReadLine();

if (input == "b")

{

return;

}

else if (input == "p")

{

ConfirmOrder();

return;

}

else if (input == "c")

{

CancelOrder();

return;

}

else if (input == "q")

{

Environment.Exit(0);

}

if (int.TryParse(input, out int quantity) && quantity > 0)

{

currentOrder.Items.Add(new OrderItem(currentOrder.Items.Count + 1, product, quantity));

ProductAddedToOrder();

}

else

{

Console.WriteLine("Неверный ввод, попробуйте еще раз.");

}

}

static void ProductAddedToOrder()

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Продукт успешно добавлен в заказ. Введите номер действия.");

Console.WriteLine("1. Продолжить выбор продуктов");

Console.WriteLine("[b – назад, p – оформить заказ, c - отменить заказ, q – завершить работу]");

string input = Console.ReadLine();

if (input == "1")

{

SelectCategory();

}

else if (input == "b")

{

SelectCategory();

}

else if (input == "p")

{

ConfirmOrder();

}

else if (input == "c")

{

CancelOrder();

}

else if (input == "q")

{

Environment.Exit(0);

}

}

static void ConfirmOrder()

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Ваш заказ сформирован.");

decimal total = 0;

foreach (var item in currentOrder.Items)

{

decimal itemTotal = item.Product.Price \* item.Quantity;

total += itemTotal;

Console.WriteLine($"{item.Product.Name} x {item.Quantity} = {itemTotal} руб.");

}

Console.WriteLine($"Итого = {total} руб.");

Console.WriteLine("Выберите действие:");

Console.WriteLine("[b – назад, a – подтвердить заказ, c - отменить заказ, q – завершить работу]");

string input = Console.ReadLine();

if (input == "b")

{

SelectCategory();

}

else if (input == "a")

{

FinalizeOrder();

}

else if (input == "c")

{

CancelOrder();

}

else if (input == "q")

{

Environment.Exit(0);

}

}

static void FinalizeOrder()

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Ваш заказ оформлен!");

context.Orders.Add(currentOrder);

currentOrder = null;

Console.WriteLine("[m – в меню, q – завершить работу]");

string input = Console.ReadLine();

if (input == "m")

{

MainMenu();

}

else if (input == "q")

{

Environment.Exit(0);

}

}

static void CancelOrder()

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Ваш заказ отменен!");

currentOrder = null;

Console.WriteLine("[m – в меню, q – завершить работу]");

string input = Console.ReadLine();

if (input == "m")

{

MainMenu();

}

else if (input == "q")

{

Environment.Exit(0);

}

}

static void ViewOrders()

{

Console.Clear();

Console.WriteLine("Терминал для заказа.");

Console.WriteLine("Список заказов:");

var table = new ConsoleTable("Номер заказа", "Место", "Дата", "Продукты", "Цена");

foreach (var order in context.Orders)

{

var products = string.Join(", ", order.Items.Select(item => $"{item.Product.Name} x {item.Quantity}"));

table.AddRow(order.OrderNumber, order.Place, order.CreationDate.ToString("g"), products, order.Items.Sum(item => item.Product.Price \* item.Quantity));

}

table.Options.EnableCount = false;

table.Write(Format.MarkDown);

Console.WriteLine("[m – в меню, q – завершить работу]");

string input = Console.ReadLine();

if (input == "m")

{

MainMenu();

}

else if (input == "q")

{

Environment.Exit(0);

}

}

static string GenerateOrderNumber()

{

var lastOrder = context.Orders.OrderByDescending(o => o.Id).FirstOrDefault();

if (lastOrder != null && lastOrder.OrderNumber != null)

{

string lastOrderNumber = lastOrder.OrderNumber;

int lastNumber = int.Parse(lastOrderNumber.Split('-')[1]);

int newNumber = (lastNumber + 1) % 100;

return $"A-{newNumber:00}";

}

else

{

return "A-00";

}

}

}

}

# 2.5 Тестирование программы

Запускаем программу, после чего появляется данный список действий.

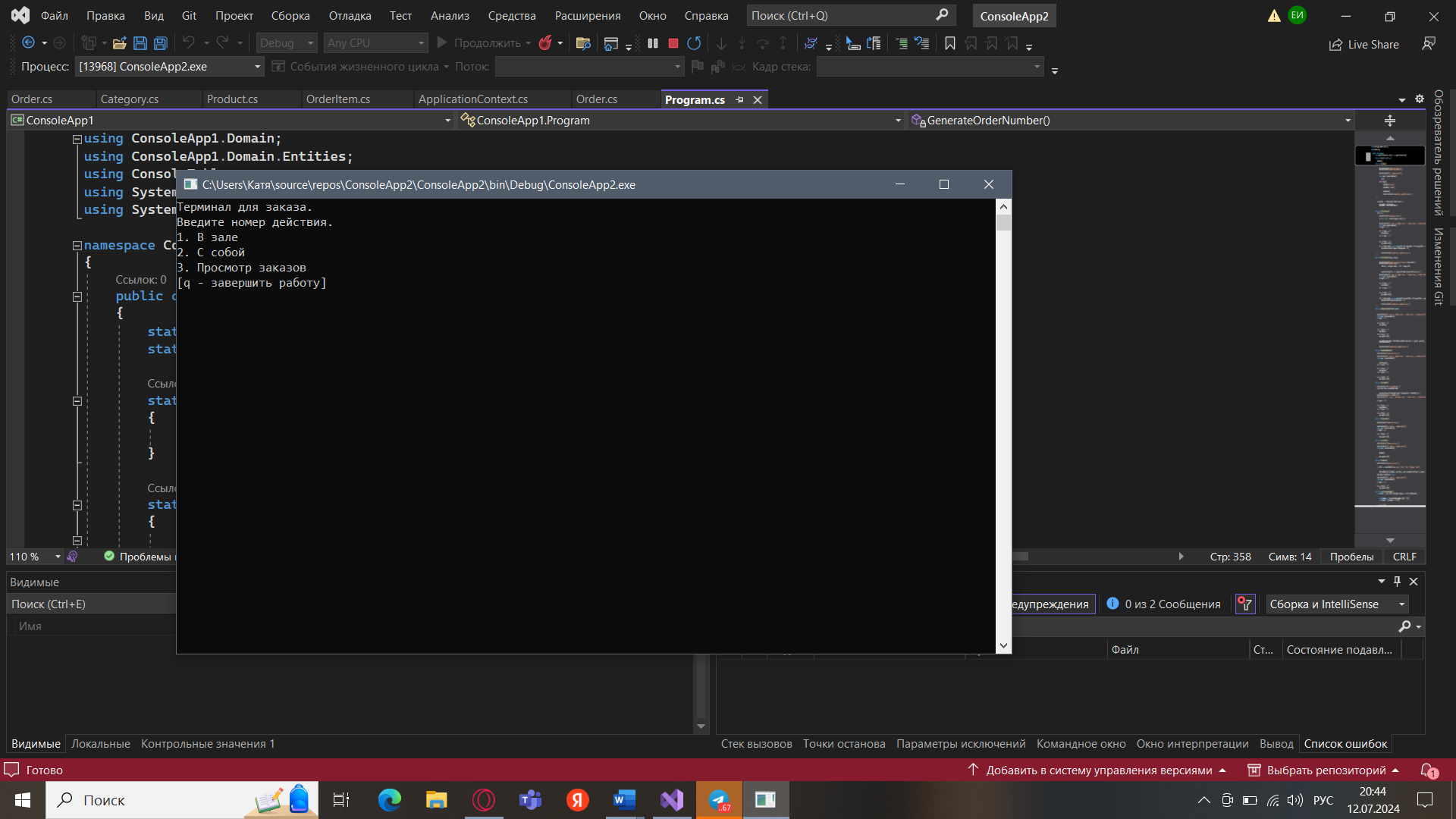


Рисунок 2.6 – Список действий

Далее появляется список имеющихся категорий блюд.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 2.7 – Список категорий

После выбора категории «Пицца» в окне появляется информация о блюдах, а также их стоимость.

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Рисунок 2.8 – Список блюд категории «Пицца»

Аналогично с другими категориями. После выполнения заказа, и добавления соответствующих элементов пользователь может либо добавить ещё продукты в заказ, либо завершить его. При завершении выводится соответствующее сообщение.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2.9 – Просмотр заказа

# Заключение

Таким образом, создание информационной системы по учету заказов в ресторане позволит значительно упростить и автоматизировать процесс работы персонала, улучшить сервис для посетителей, и в итоге повысить эффективность и прибыльность бизнеса. Разработка такой системы требует внимательного анализа потребностей ресторана, грамотного планирования всех этапов и вовлечения специалистов с опытом в данной области. В результате создания информационной системы по учету заказов ресторан сможет значительно улучшить свою конкурентоспособность и обеспечить более качественное обслуживание клиентов.

# Список литературы

1. Влияние информационных технологий в общественном питании URL: https://studylib.ru/doc/6338037/vliyanie-informacionnyh-tehnologij-na-predpriyatiya-obshhestve
2. Информационные технологии в общественном питании URL: https://spravochnick.ru/Informaciones\_tehnologii/informacionnye\_tehnologii\_v\_obschestvennom\_pitanii/
3. Эффектность внедрения информационных технологий на предприятия питания URL: https://cyberleninka.ru/article/n/effektivnost-vnedreniya-informatsionnyh-tehnologiy-na-predpriyatii-pitaniya
4. Хабр / C# и разработка консольных приложений: [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/ruvds/blog/482464/ – 8.07.2023
5. Язык программирования C# URL: https://timeweb.com/ru/community/articles/chto-takoe-csharp
6. IT-технологии в сфере общественного питания URL: https://pc.uz/news/151860-it-tehnologii-v-sfere-obshestvennogo-pitaniya
7. Информационные технологии в ОП [Электронный ресурс]: портал. – Электрон. дан. – Режим доступа: https://infourok.ru/informacionnietehnologii-v-obschestvennom-pitanii-dlya-i-kursa-2075595.html.
8. LinkedIn Learning / Learning C# (2023): [Электронный ресурс]. – Режим доступа: https://www.linkedin.com/learning/learning–c–sharp–2023/ – 8.07.2023