МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по производственной практике**

**ПМ01. Разработка модулей программного обеспечения для компьютерных систем**

**Тема: «Разработка программного модуля «Кофейня»»**

Студент

*Поглазова Валерия Владимировна*

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*Калинин Арсений Олегович*

Руководитель практики от организации:

*Павленко Сергей Михайлович*

подпись

УТВЕРЖДАЮ:

Директор

*Устюжанинов Вадим Борисович*

Наименование организации

ООО "Электрон-Софт"

\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка

М. П.

2024-2025 уч. год

**Содержание**

[1.СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ (БАЗЕ ПРАКТИКИ) 3](#_Toc195946976)

[2. РАЗРАБОТКА И АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ 4](#_Toc195946977)

[3. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПО АЛГОРИТМУ В СООТВЕТСТВИИ ТЕХНИЧЕСКОМУ ЗАДАНИЮ 8](#_Toc195946978)

[4. ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ МОДУЛЬ 14](#_Toc195946979)

[5. ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНОГО МОДУЛЯ 24](#_Toc195946980)

[6. ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ 26](#_Toc195946981)

[7. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ 29](#_Toc195946982)

[8. РЕФАКТОРИНГ И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА 30](#_Toc195946983)

[9. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА 32](#_Toc195946984)

[10. ЗАКЛЮЧЕНИЕ 36](#_Toc195946985)

[ПРИЛОЖЕНИЯ 37](#_Toc195946986)

# СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ (БАЗЕ ПРАКТИКИ)

Производственную практику по модулю ПМ.01. «Разработка модулей программного обеспечения для компьютерных систем» я проходила в компании ООО «Электрон-Софт»:

Юридический адрес: 610004, Кировская область, г. Киров, ул. Ленина, д. 68.

ФИО руководителя (директора) – Устюжанинов Вадим Борисович

Телефон: +7 (8332) 71-49-10

* Специализация:

ООО "Электрон-Софт" является компанией, специализирующейся на выполнение работ и оказание услуг в сфере 1С.

* Структура:

Управление:

* Директор
* Генеральный директор
* Заместитель директора
* Заместитель генерального директора

В первый день прохождения данного модуля нам провели инструктаж по технике безопасности и провели онлайн-тестирование

# 2. РАЗРАБОТКА И АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Прежде чем приступить к разработке программного модуля, необходимо детально изучить требования и спроектировать архитектуру системы. Этот этап включил анализ технического задания, создание диаграмм, которые стали основой для дальнейшей работы.

На этапе разработки и анализа технического задания было выполнено следующие:

1. На основе предоставленного технического задания были выделены ключевые функциональные и нефункциональные требования к системе. Определены основные сущности, процессы и роли пользователей.
2. Были созданы диаграммы, описывающие структуру и поведение системы.

• Диаграмма вариантов использования (Use Case Diagram) отображает основные сценарии взаимодействия пользователей (Администратора, Бариста, Менеджера и Официанта) с системой (Рисунок 2.1).



Рисунок 2.1 – Диаграмма вариантов использования

• Диаграмма последовательности (Sequence Diagram) показывает последовательность формирования заказа и выполнение заказа (Рисунок 2.2).

Рисунок 2.2 – Диаграмма последовательности

• Диаграмма видов деятельности (Activity Diagram) (Рисунок 2.3).

Диаграмма иллюстрирует поток работы в системе, такой как процесс приемки заказа.

• ER-диаграмма (Entity-Relationship Diagram).

Диаграмма отображает структуру базы данных, включая сущности, их атрибуты и связи между ними (Рисунок 2.4).

3. На основе ER-диаграммы была спроектирована база данных, реализованная в MS SQL Server (Рисунок 2.5).

В результате анализа технического задания были созданы диаграммы, которые легли в основу разработки программного модуля. Использование программы Microsoft Visio позволило эффективно визуализировать структуру и поведение системы, что значительно упростило процесс проектирования.



Рисунок 2.3 – Диаграмма видов деятельности



Рисунок 2.4 – ER-диаграмма

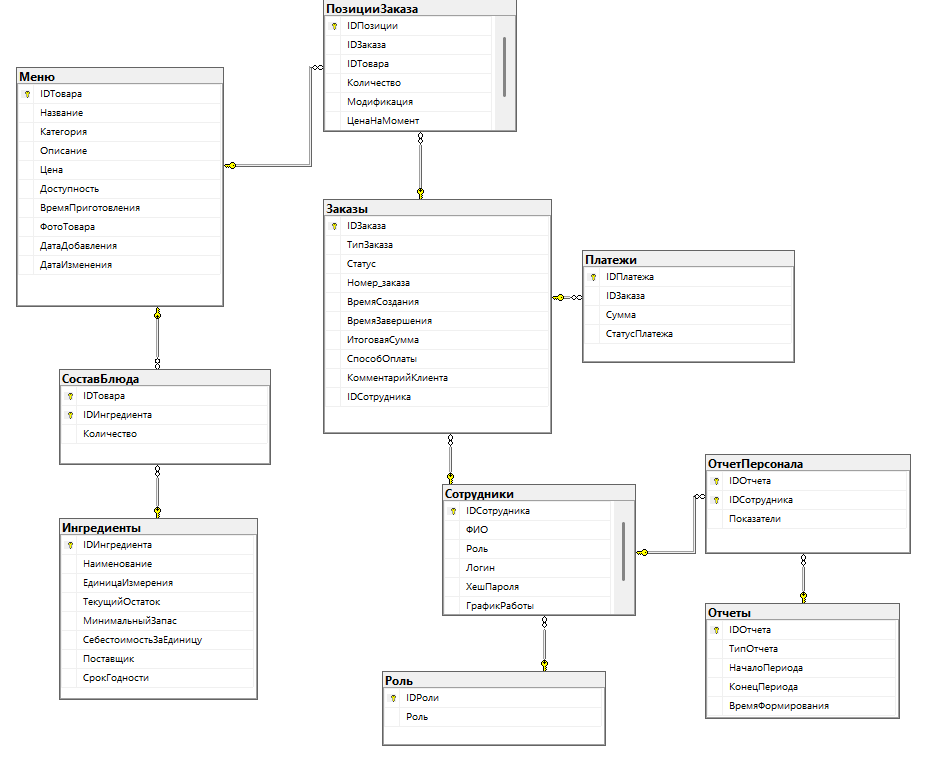


Рисунок 2.5 – Диаграмма базы данных

# 3. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПО АЛГОРИТМУ В СООТВЕТСТВИИ ТЕХНИЧЕСКОМУ ЗАДАНИЮ

На данном этапе была выполнена разработка программного модуля, включающего в себя библиотеку для работы с данными и настольное приложение, в строгом соответствии с техническим заданием (ТЗ). Основной задачей было реализовать функционал, который обеспечивает корректную работу системы.

**Разработка API**

API разработано с необходимым функционалом по техническому заданию. Реализованы методы GET и POST запросов (Рисунок 6):

1. Авторизация (2 способа).

2. Получение списка меню.

3. Получение списка заказов.

4. Отправка данных о товаре.

Основные маршруты API:

**Авторизация**

POST api/auth/login - Аутентификация пользователя (первый этап)

**Управление меню**

GET api/Menu. - Получить список всех пунктов меню.

GET api/Menu/(id) - Получить детали пункта меню по ID.

POST api/Menu - Добавить новый пункт меню (требует прав администратора).

**Управление Заказами**

GET api/Orders - Получить список заказов (для администратора или пользователя).

GET api/Orders/(id) - Получить список заказов (для администратора или пользователя).

POST api/Orders - Создать новый заказ.

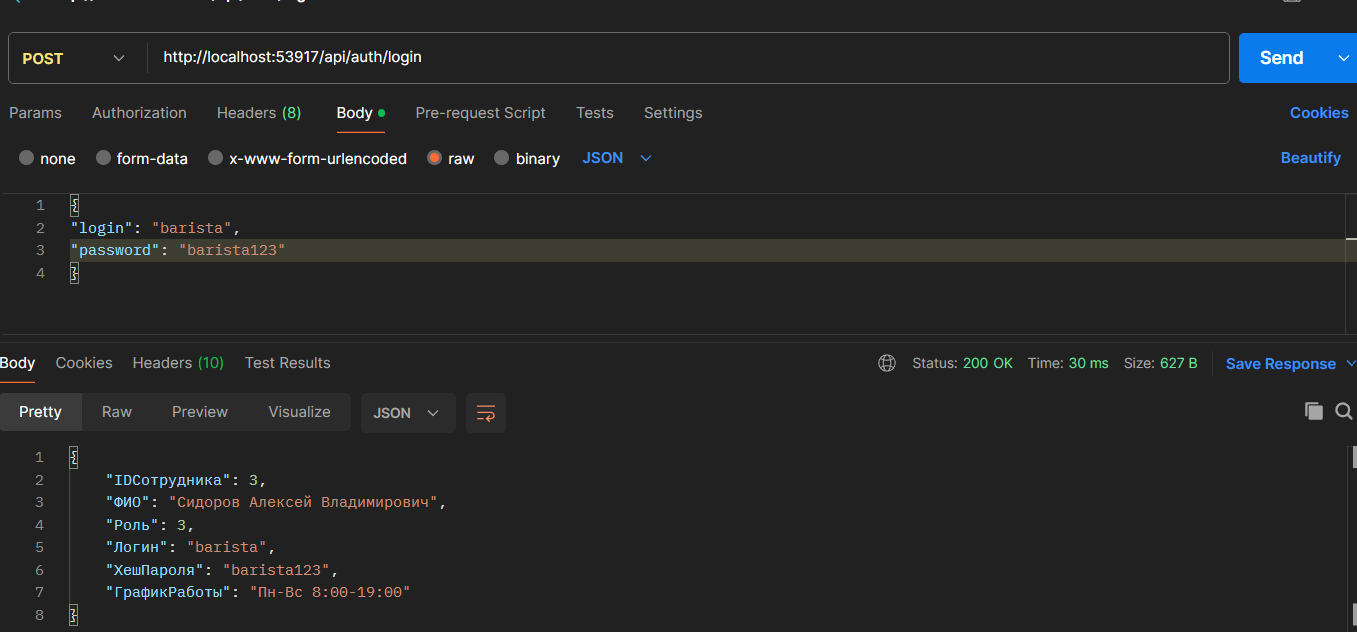


Рисунок 3.1 –API

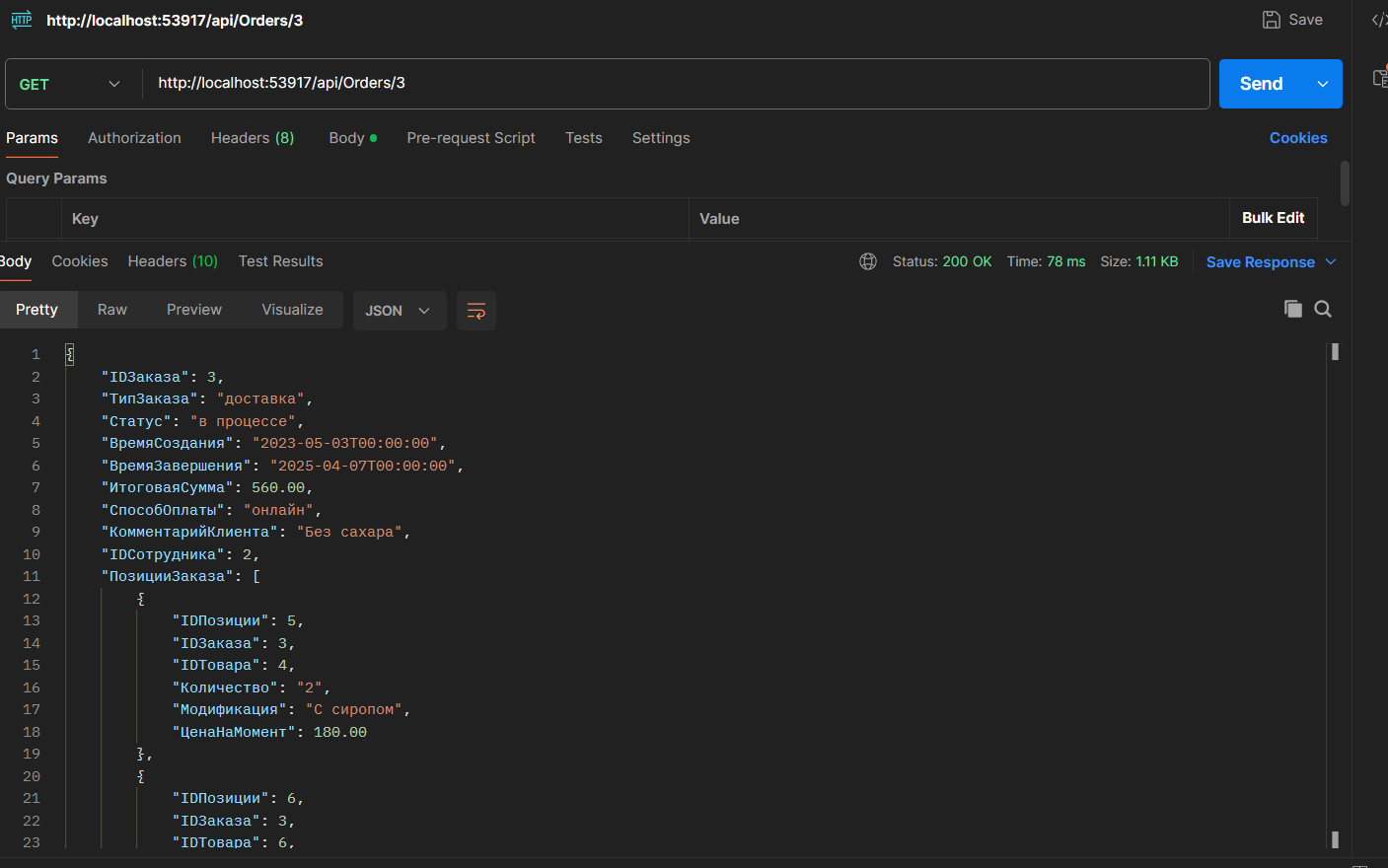


Рисунок 3.2 –API

**Разработка библиотеки классов**

Библиотека CoffeHouseLibrary разработана на языке программирования C# с использованием Entity Framework (ADO.NET Entity Data Model) для работы с базой данных кофейни. Библиотека предоставляет удобный интерфейс для управления основными бизнес-процессами.

1. Подсчет общего количества заказов (всего).

2. Подсчет суммы стоимости заказов за конкретный месяц.

3. Подсчет количества заказов определенного типа за определенный период.

4. Подсчет количества проданных товаров определенной категории за определенный период.

5. Подсчет среднего времени приготовления заказов.

6. Подсчет остатков ингредиентов ниже минимального запаса.



Рисунка 3.3 – Библиотека

На рисунке 3.4 изображен методы библиотеки классов.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  namespace CoffeeHouseLibrary  {  public class CoffeeHouseManager  {  private readonly CoffeeHouseEntities \_context;  public CoffeeHouseManager()  {  \_context = new CoffeeHouseEntities();  }  // 1. Подсчет общего количества заказов (всего)  public int GetTotalOrdersCount()  {  return \_context.Заказы.Count();  }  // 2. Подсчет суммы стоимости заказов за конкретный месяц  public decimal GetMonthlyRevenue(int year, int month)  {  var startDate = new DateTime(year, month, 1);  var endDate = startDate.AddMonths(1).AddDays(-1);  return \_context.Заказы  .Where(z => z.ВремяСоздания >= startDate && z.ВремяСоздания <= endDate)  .Select(z => z.ИтоговаяСумма ?? 0m)  .Sum();  }  // 3. Подсчет количества заказов определенного типа за определенный период  public int GetOrdersCountByType(string orderType, DateTime startDate, DateTime endDate)  {  return \_context.Заказы  .Count(z => z.ТипЗаказа == orderType && z.ВремяСоздания >= startDate && z.ВремяСоздания <= endDate);  }  // 4. Подсчет количества проданных товаров определенной категории за определенный период  public int GetSoldItemsByCategory(string category, DateTime startDate, DateTime endDate)  {  return \_context.ПозицииЗаказа  .Where(pz => pz.Заказы.ВремяСоздания >= startDate && pz.Заказы.ВремяСоздания <= endDate && pz.Меню.Категория == category)  .Sum(pz => SafeConvertToInt(pz.Количество));  }  // 5. Подсчет среднего времени приготовления заказов  public TimeSpan GetAverageOrderPreparationTime()  {  var averageTicks = \_context.Заказы  .Where(z => z.ВремяЗавершения != null)  .Select(z => (z.ВремяЗавершения - z.ВремяСоздания).Value.Ticks)  .DefaultIfEmpty(0)  .Average();  return TimeSpan.FromTicks((long)averageTicks);  }  // 6. Подсчет остатков ингредиентов ниже минимального запаса  public int GetLowStockIngredientsCount()  {  return \_context.Ингредиенты  .Count(i => SafeConvertToInt(i.ТекущийОстаток) < SafeConvertToInt(i.МинимальныйЗапас));  }  // Вспомогательный метод для безопасного преобразования строки в целое число  private int SafeConvertToInt(string value)  {  if (int.TryParse(value, out int result))  {  return result;  }  return 0;  }  }  } |

Рисунка 3.4 – Методы библиотеки классов

**Разработка настольного приложения**

Настольное приложение было разработано для администратора клиники с использованием WPF для C#. Основные функции приложения:

* Двухфакторную аутентификацию и разграничение по ролям: Администратор, Менеджера, Официанта, Бариста.
* Управление заказами, меню и пользователями.
* Просмотр отчетов, настройка системы.
* Контроль заказов, заявки поставщикам, формирование отчетов
* Отслеживание заказов, изменение статуса заказа, просмотр журнала приготовлений
* Поиск и сортировка данных: заказы, накладные, товары.
* Управление пользователями: редактирование данных и смена пароля.
* Подключение к базе данных — приложение работает через подключение к БД по технологии ADO NET.EDM.

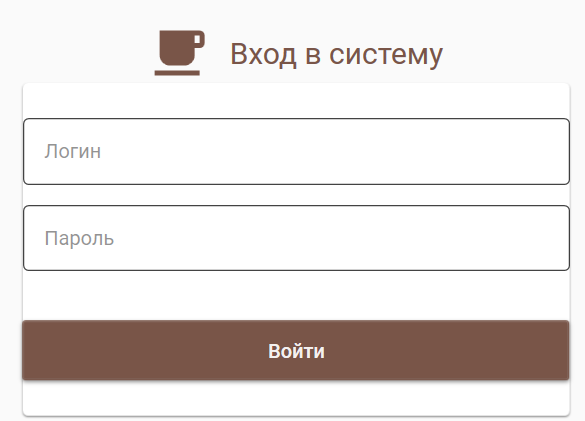


Рисунок 3.5 – Настольное приложение

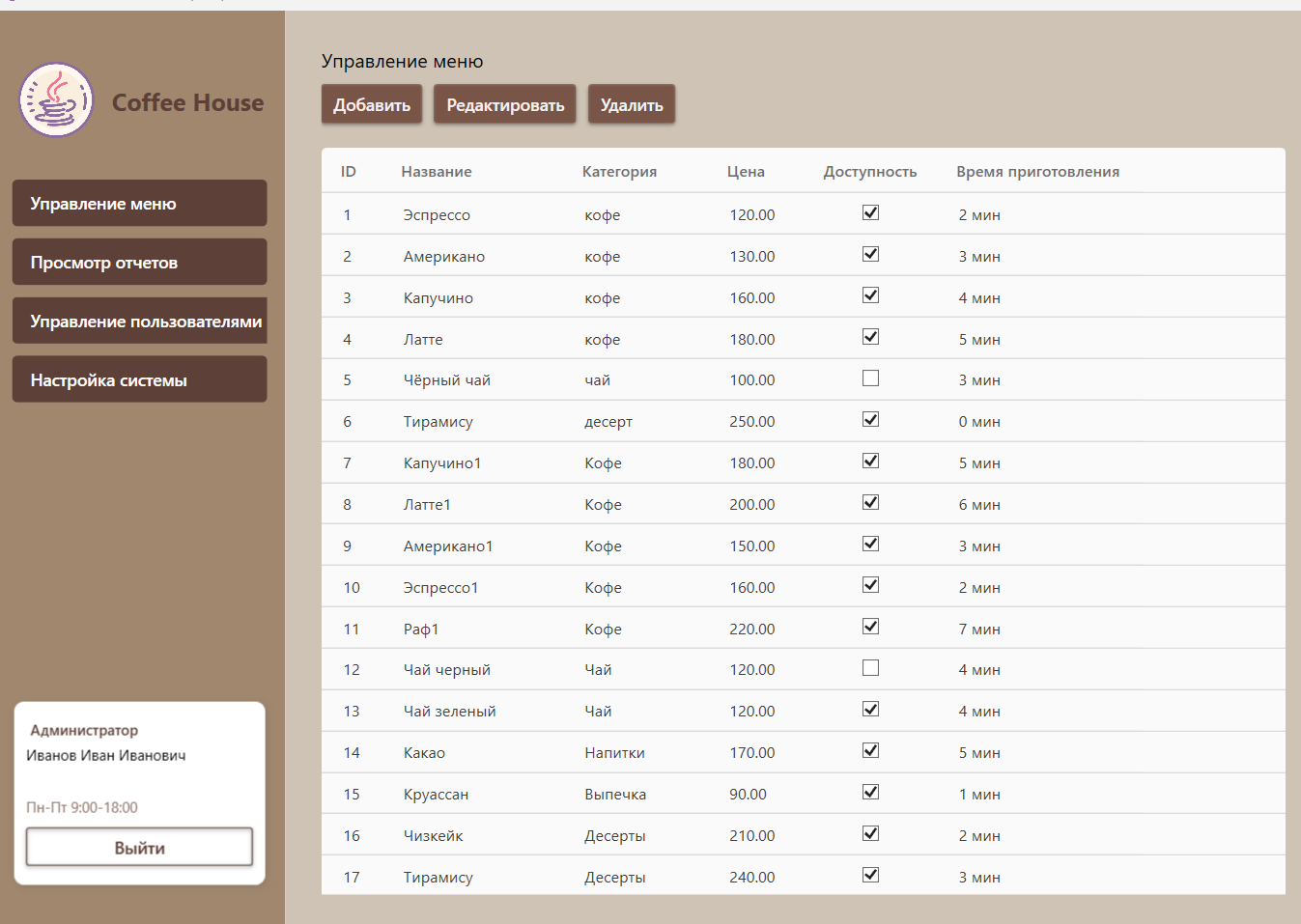
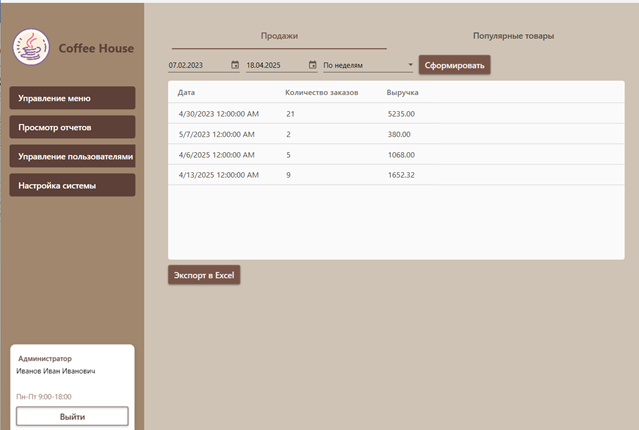


Рисунок 3.6– Настольное приложение

Рисунок 3.7– Настольное приложение

На рисунке 3.8 изображена часть структуры настольного приложения.

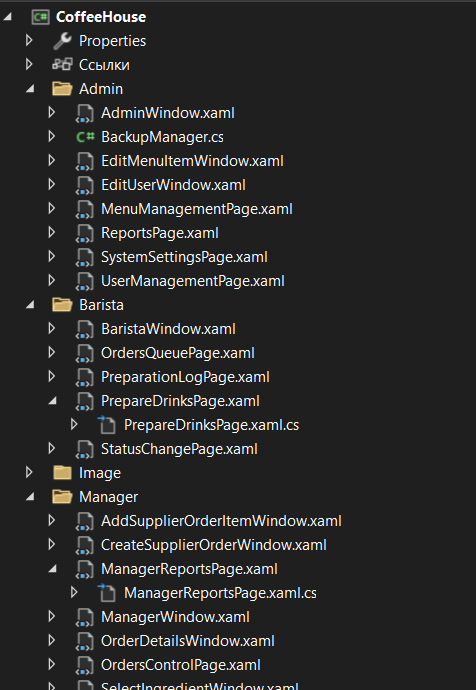


Рисунок 3.8 – Структура приложения

# 4. ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ МОДУЛЬ

Качественная документация является важной частью любого программного продукта. Она обеспечивает удобство использования, поддержки и дальнейшего развития системы.

На данном этапе была выполнена разработка и оформление документации на программный модуль в соответствии с требованиями технического задания (ТЗ) и стандартами разработки.

**Руководство оператора для настольного приложения**

**Функциональным назначением программы** является автоматизация складских процессов.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Двухфакторную аутентификацию и разграничение по ролям: Администратор, Менеджера, Официанта, Бариста.
* Управление заказами, меню и пользователями.
* Просмотр отчетов, настройка системы.
* Контроль заказов, заявки поставщикам, формирование отчетов
* Отслеживание заказов, изменение статуса заказа, просмотр журнала приготовлений
* Поиск и сортировка данных: заказы, накладные, товары.
* Управление пользователями: редактирование данных и смена пароля.

**Условия выполнения программы**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средства в части условий их эксплуатации.

**Минимальный состав технических средств**

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор с тактовой частотой, 1 ГГц, не менее;
* оперативную память объемом, 512 Мб, не менее;
* жесткий диск со свободным местом 500 Мб, не менее;
* монитор, с разрешением экрана 1024\*768, не менее;
* компьютерная мышь;
* клавиатура;
* принтер;
* CD – привод.

**Минимальный состав программных средств**

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7 и выше.

**Требования к персоналу (пользователю)**

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – пользователь программы, оператор.

Пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы семейства Windows.

**Выполнение программы**

Для установки программы необходимо открыть файл setup.exe от имени администратора с CD–диска. При установке выбрать установку для всех пользователей. После установки ярлыки приложений отобразиться на рабочем столе. Для запуска клиентской программы необходимо открыть ярлык либо файл Автоматизация складского учета.exe в папке установки, после пользователю отобразится окно авторизации программы (Рисунок 1).

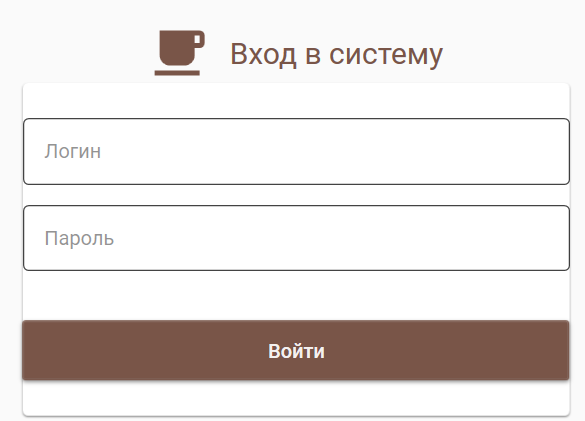


Рисунок 4.1 – Окно авторизации

Для того что бы начать использовать приложение вам необходимо авторизоваться. Для этого, необходимо ввести в поля Логин и Пароль соответствующие данные и нажать на кнопку «Войти» и производится переход на окно с нужным функционалом, соответствующим вашей роли в системе: Администратор, Менеджера, Официанта, Бариста.

**Функционал Администратора**

Для администратора доступны следующие функции:

Администратор имеет возможность управлять меню (Рисунок 4.2), совершать просмотр отчетов (Рисунок 4.3), управлять пользователями (Рисунок 4.4) и настраивать систему (Рисунок 4.5).

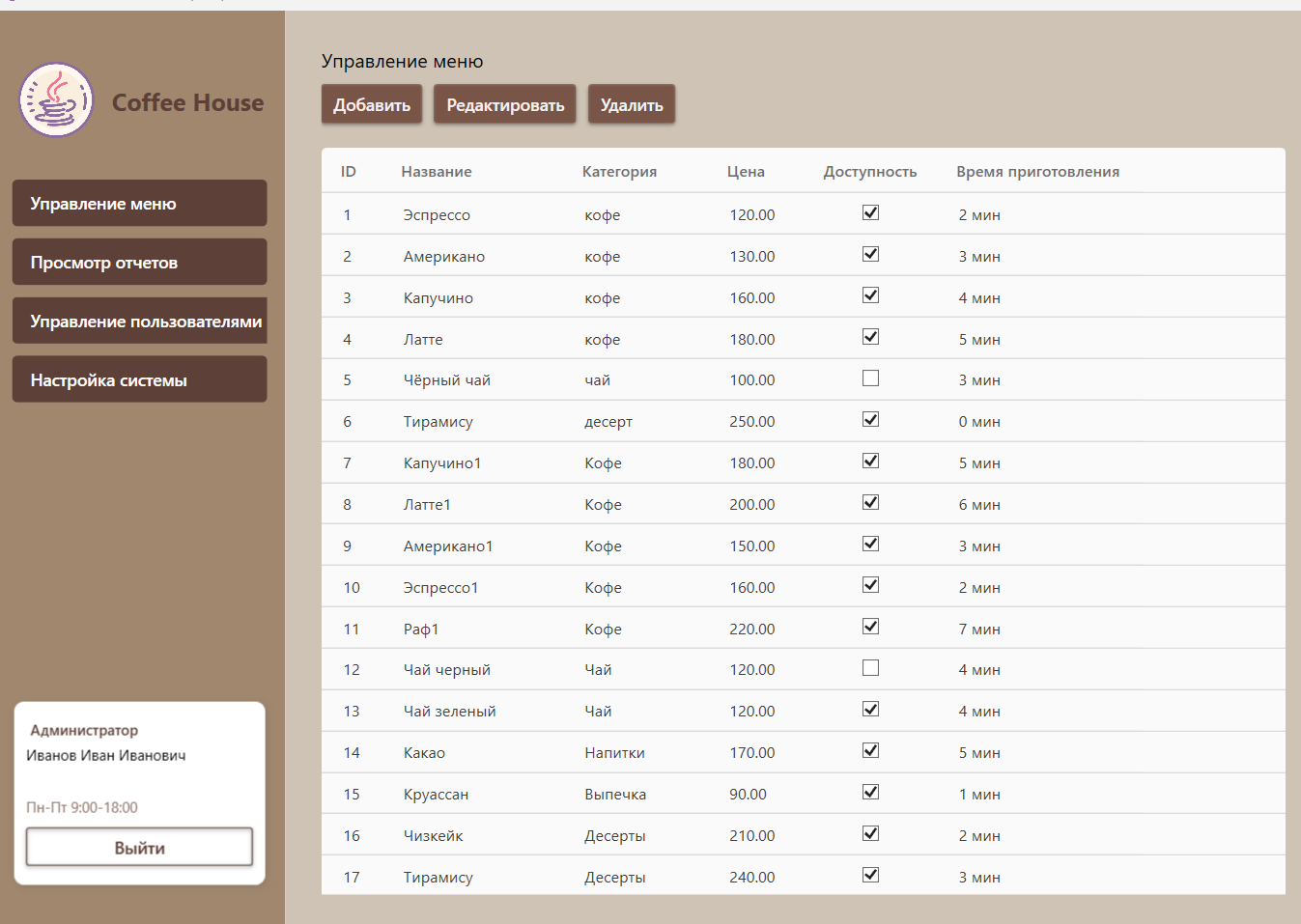
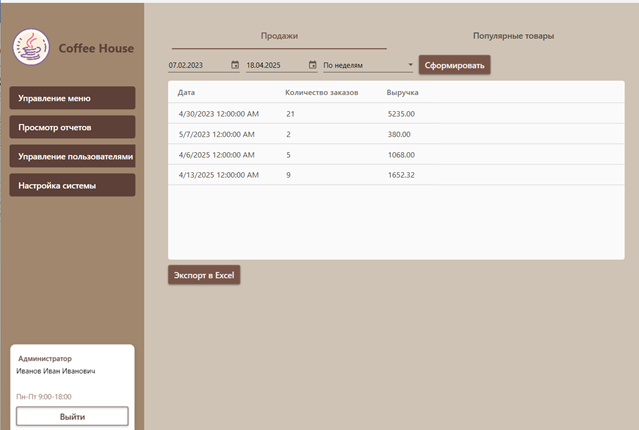


Рисунок 4.2– Окно администратора «Управление меню»

Рисунок 4.3– Окно администратора «Просмотр отчетов»

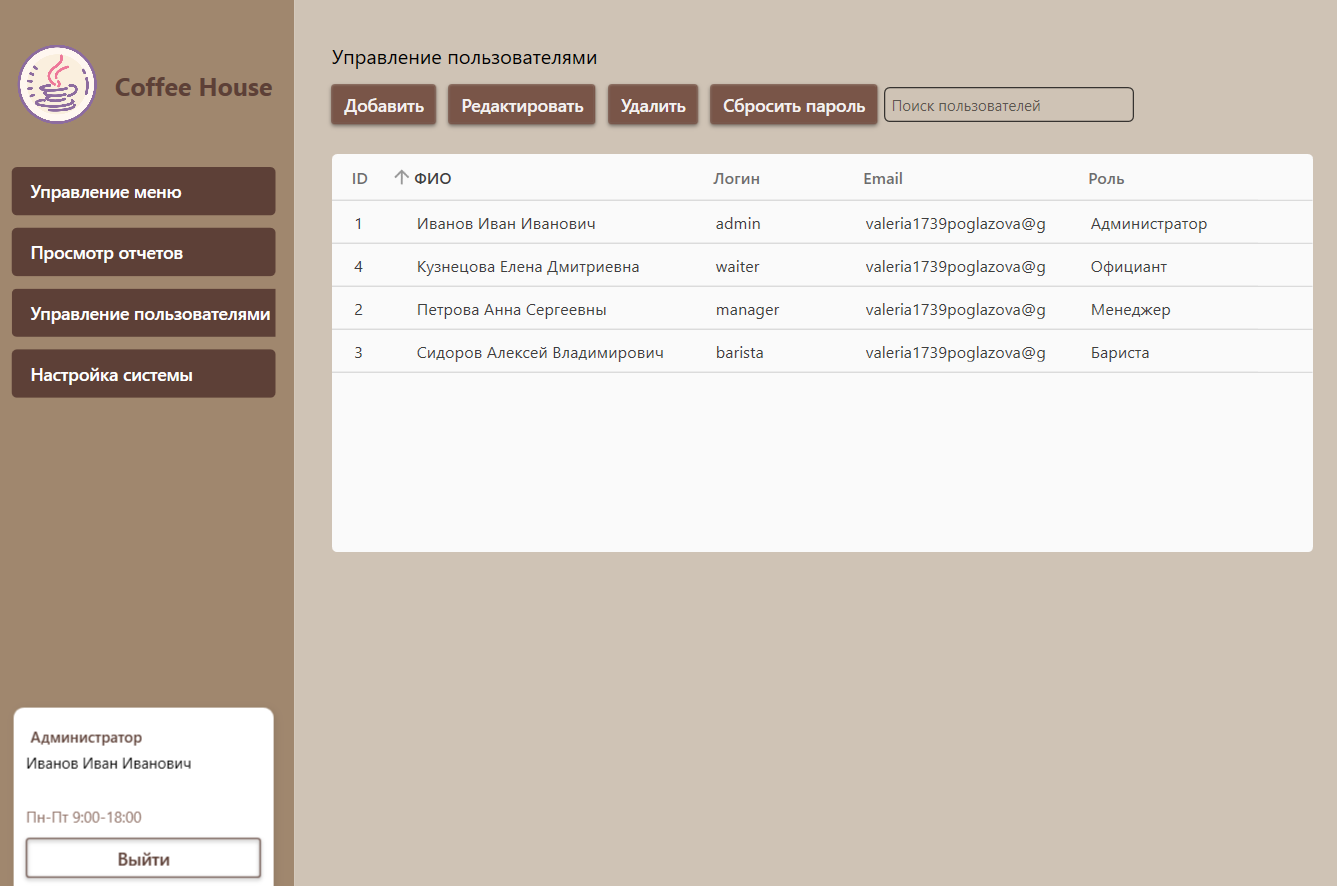


Рисунок 4.3– Окно администратора «Управление пользователями»

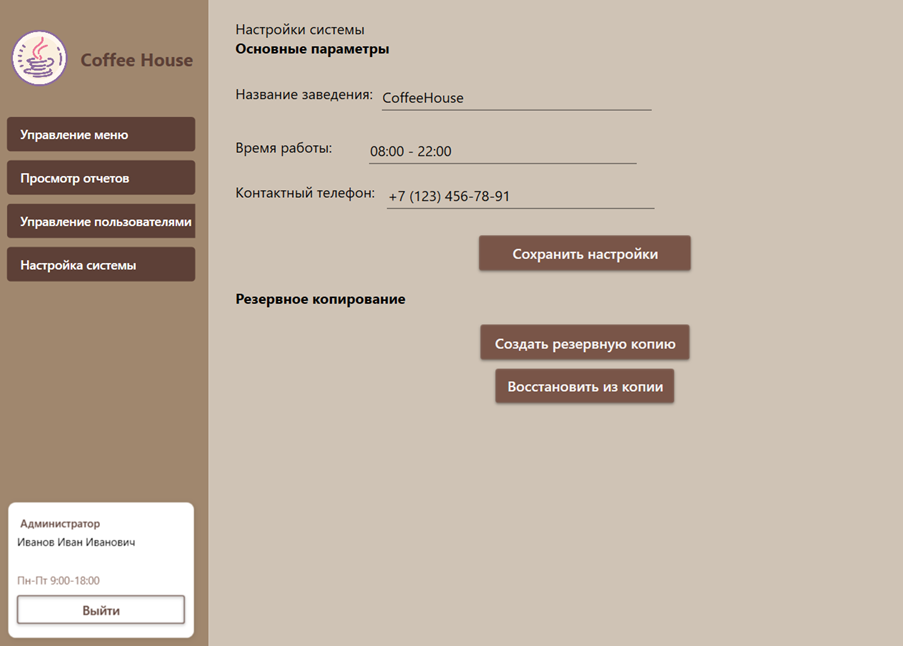


Рисунок 4.4– Окно администратора «Настройка системы»

**Функционал Менеджера**

Для менеджера доступны следующие функции:

Менеджер имеет возможность контролировать заказы (Рисунок 4.5), формировать заявки поставщикам (Рисунок 4.6), управлять меню (Рисунок 4.7), формировать отчеты (Рисунок 4.8).

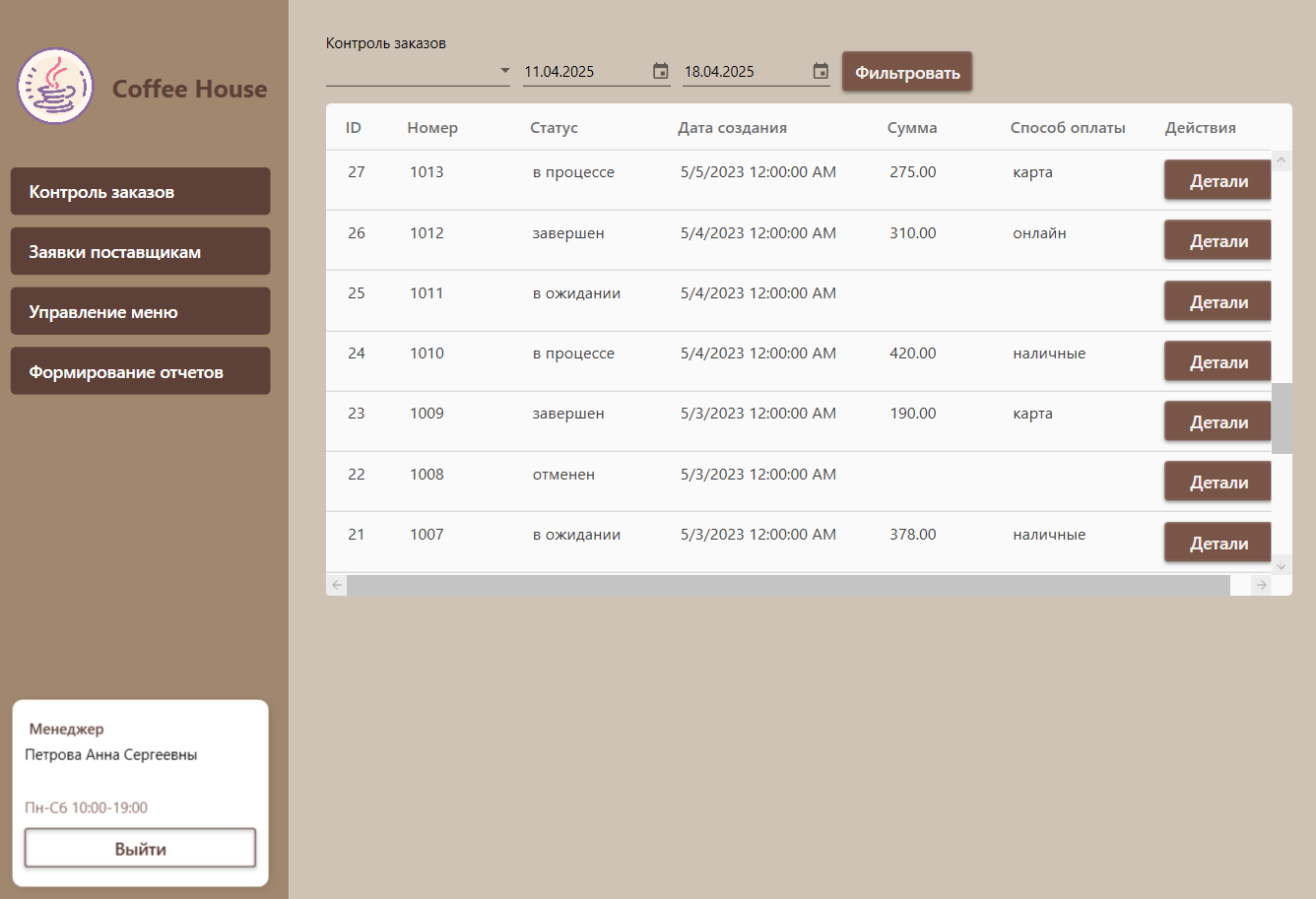


Рисунок 4.5– Окно менеджера «Настройка системы»

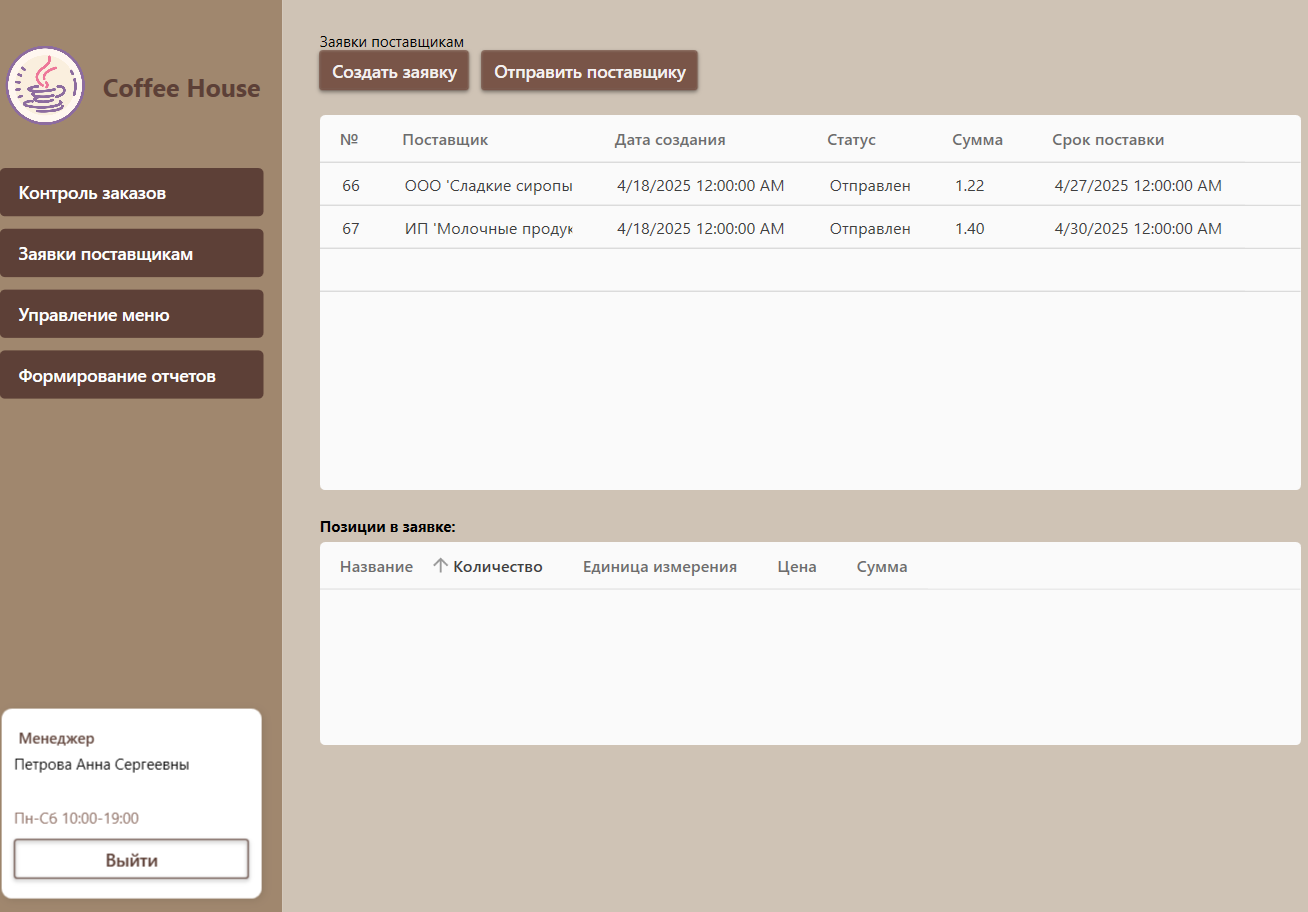


Рисунок 4.6– Окно менеджера «Заявки поставщикам»

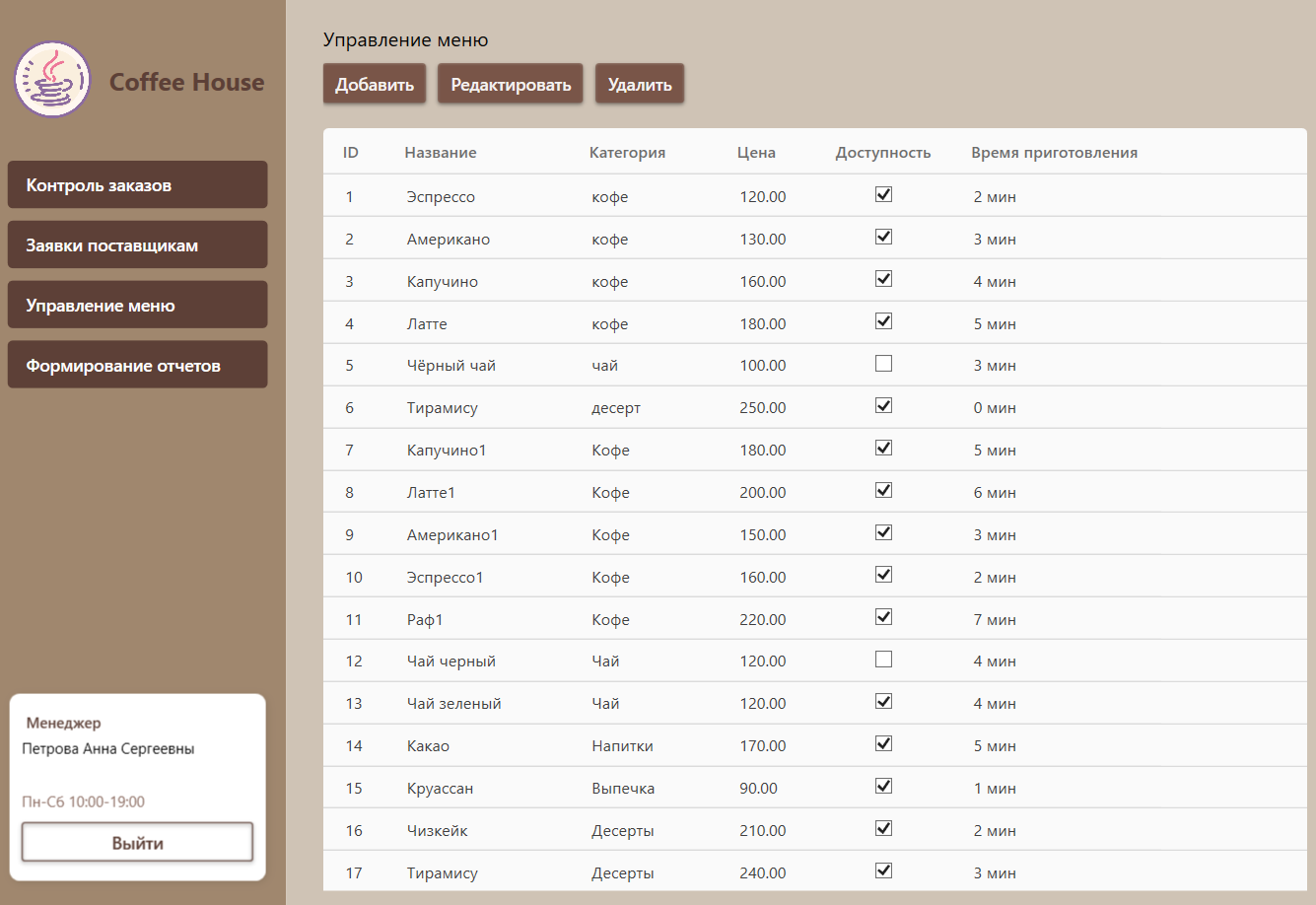


Рисунок 4.7– Окно менеджера «Управление меню»

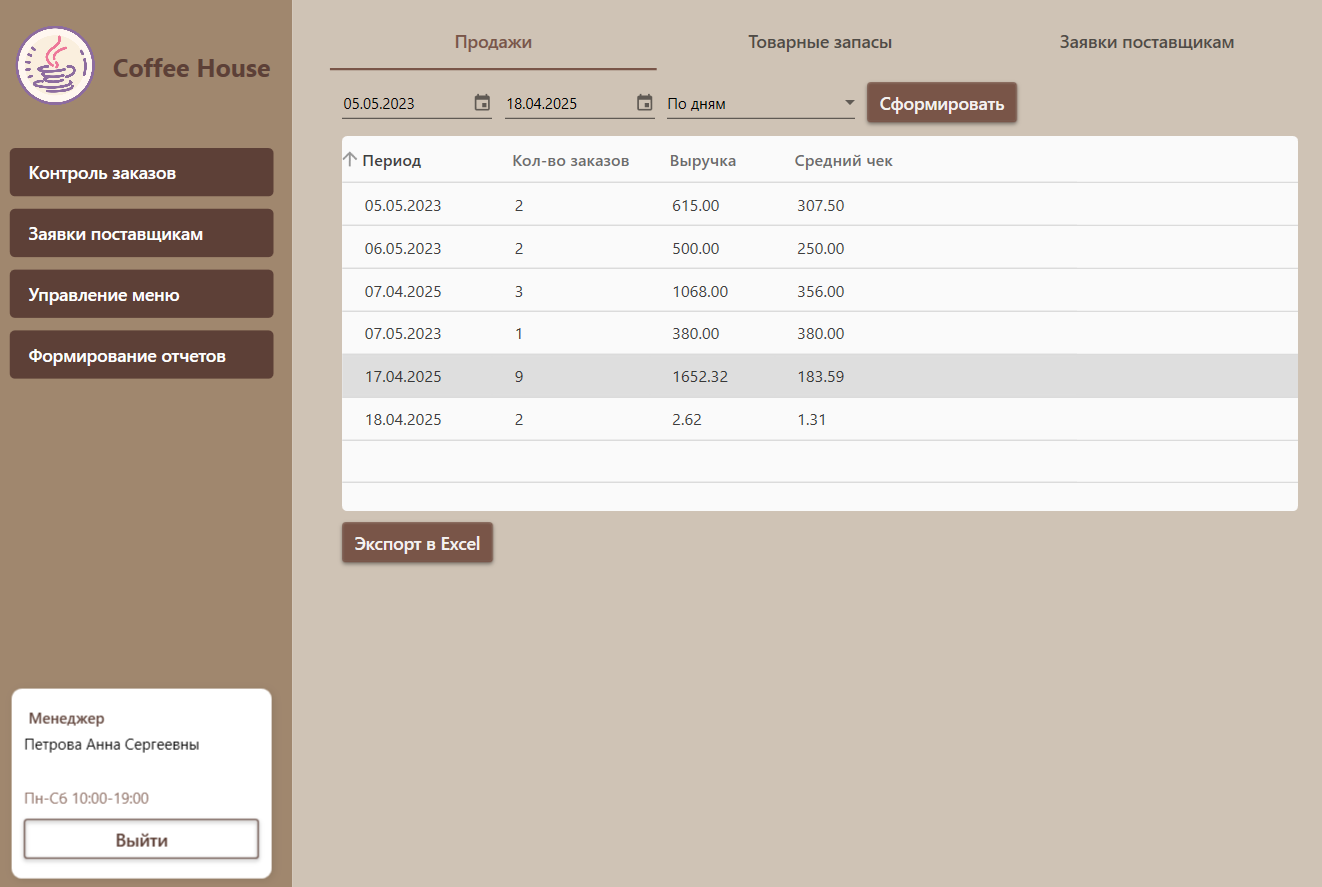


Рисунок 4.9– Окно менеджера «Формирование отчетов»

**Функционал Бариста**

Для бариста доступны следующие функции:

Бариста имеет возможность смотреть текущие заказы и меню (Рисунок 5.0), просматривать очередь заказов (Рисунок 5.1), изменять статус приготовления (Рисунок 5.2), ввести журнал приготовлений (Рисунок 5.3).

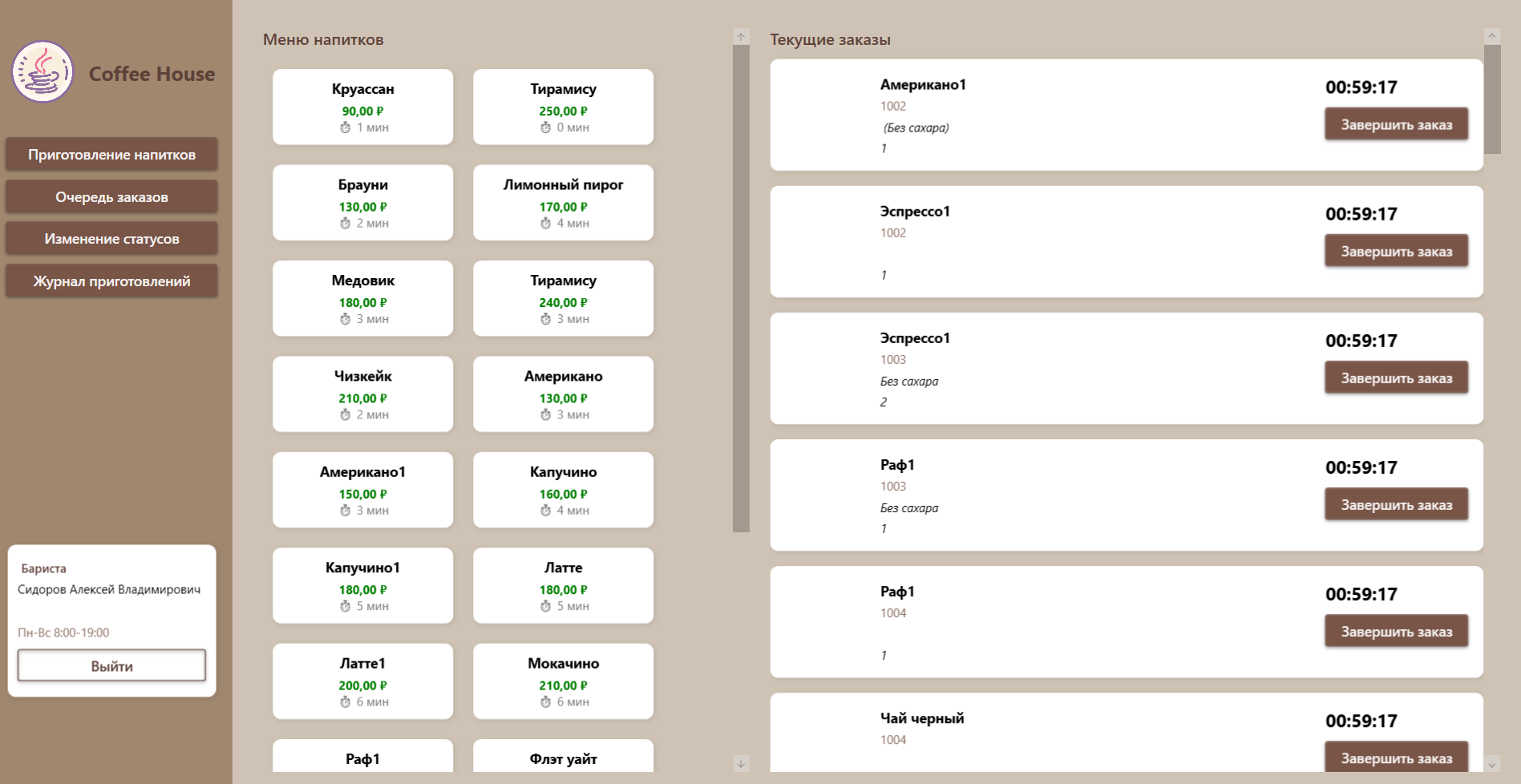


Рисунок 5.0– Окно бариста «Приготовление напитков»

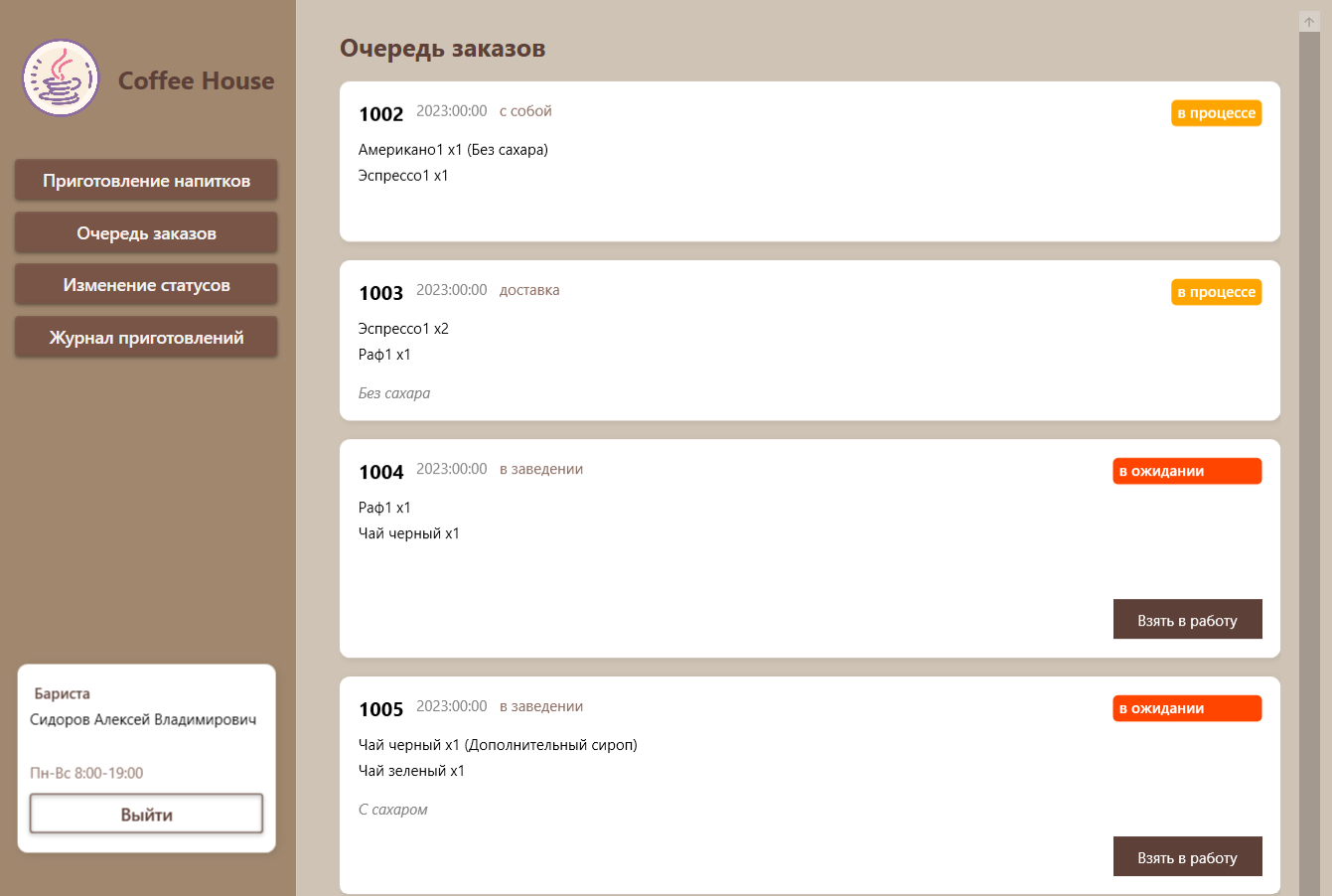


Рисунок 5.1– Окно бариста «Очередь заказов»

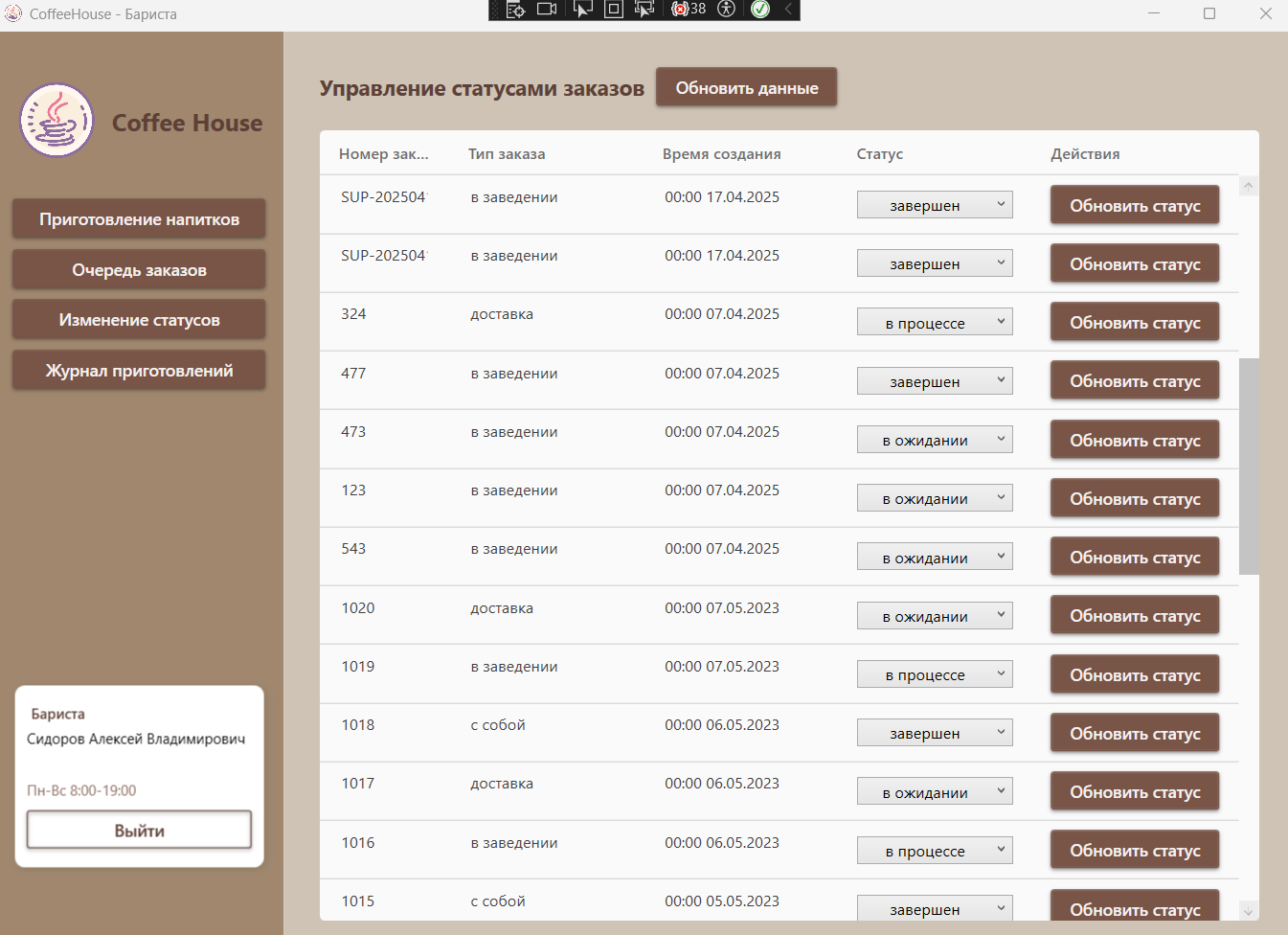


Рисунок 5.2– Окно бариста «Изменение статусов»

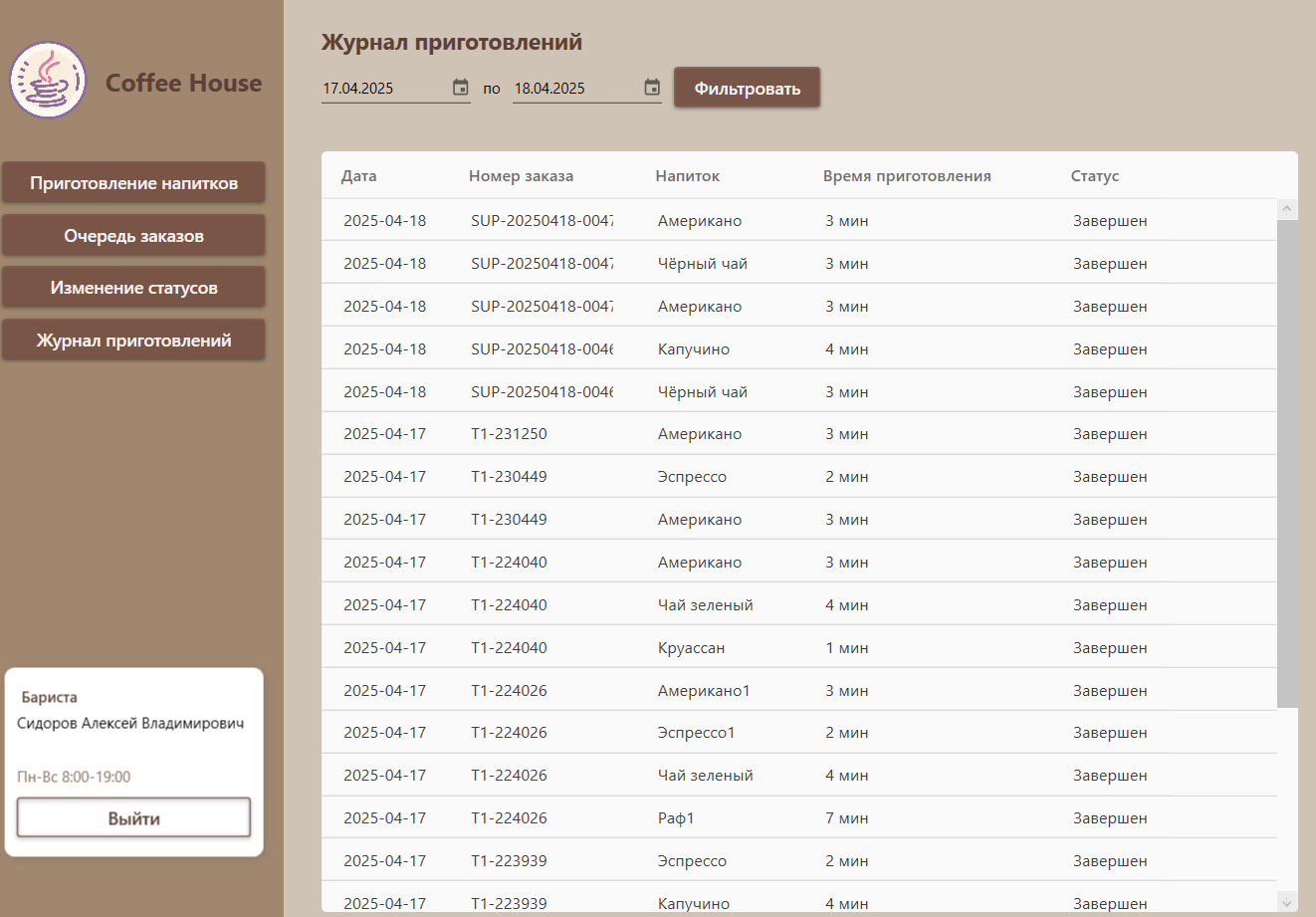


Рисунок 5.3– Окно бариста «Журнал приготовлений»

**Функционал Официанта**

Для официанта по продажам доступны следующие функции:

Официант имеет возможность принимать заказы, обслуживать клиентов, отслеживать готовность заказов (Рисунок 5.4).

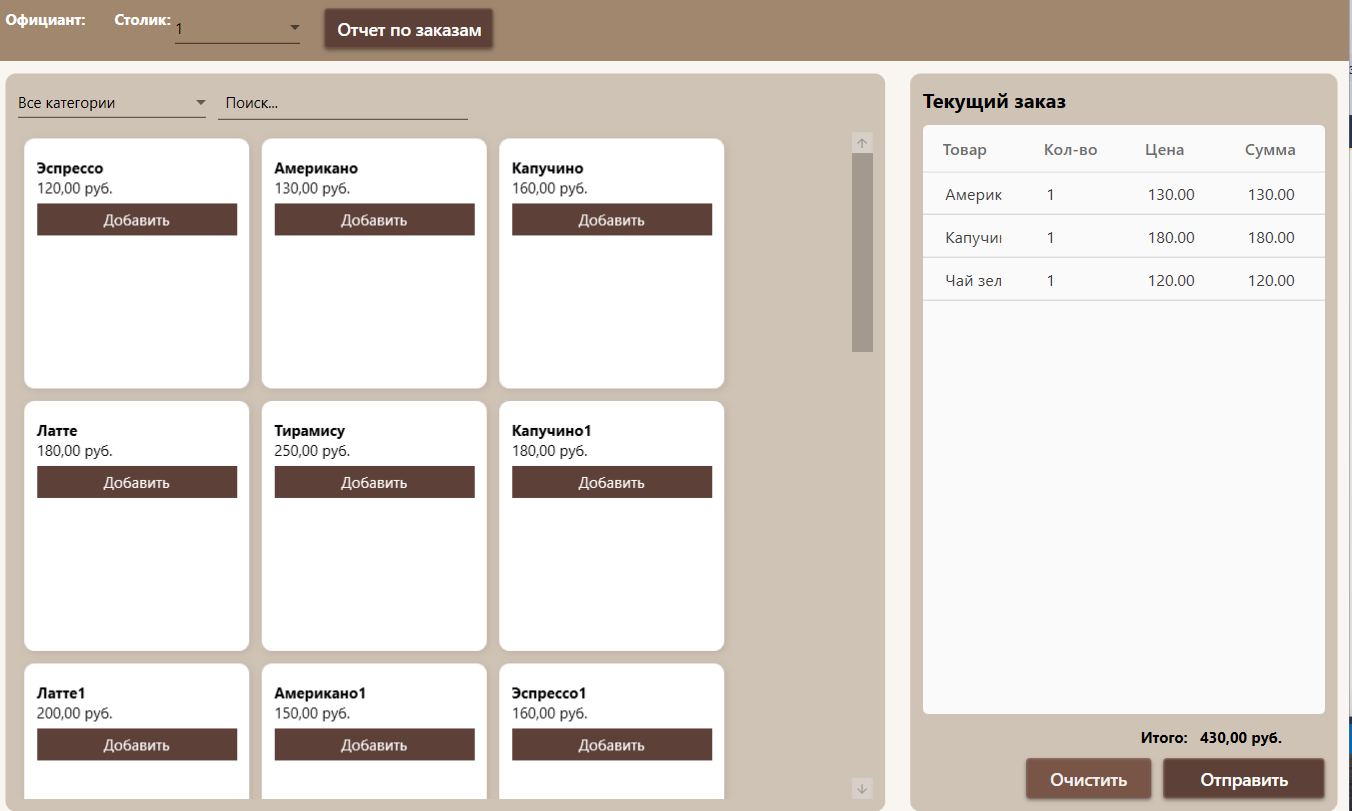


Рисунок 5.4– Окно официанта «Журнал приготовлений»

Кнопка «Выход» на панели меню позволяет вернуться к окну авторизации.

Документация оформлена в полном объеме, что обеспечивает удобство использования и поддержки программного модуля.

# 5. ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНОГО МОДУЛЯ

Отладка — это важный этап разработки, который позволяет выявить и устранить ошибки, а также оптимизировать работу системы. В данном разделе описаны основные шаги, выполненные в процессе отладки программного модуля.

На данном этапе была проведена отладка программного модуля, в ходе которой были исправлены ошибки в коде, оптимизирована работа библиотеки, api и проверена корректность функционирования настольного приложения. Основной задачей было обеспечить стабильную работу модуля в соответствии с требованиями технического задания (ТЗ).

Отладка настольного приложения (Рисунок 5.1):

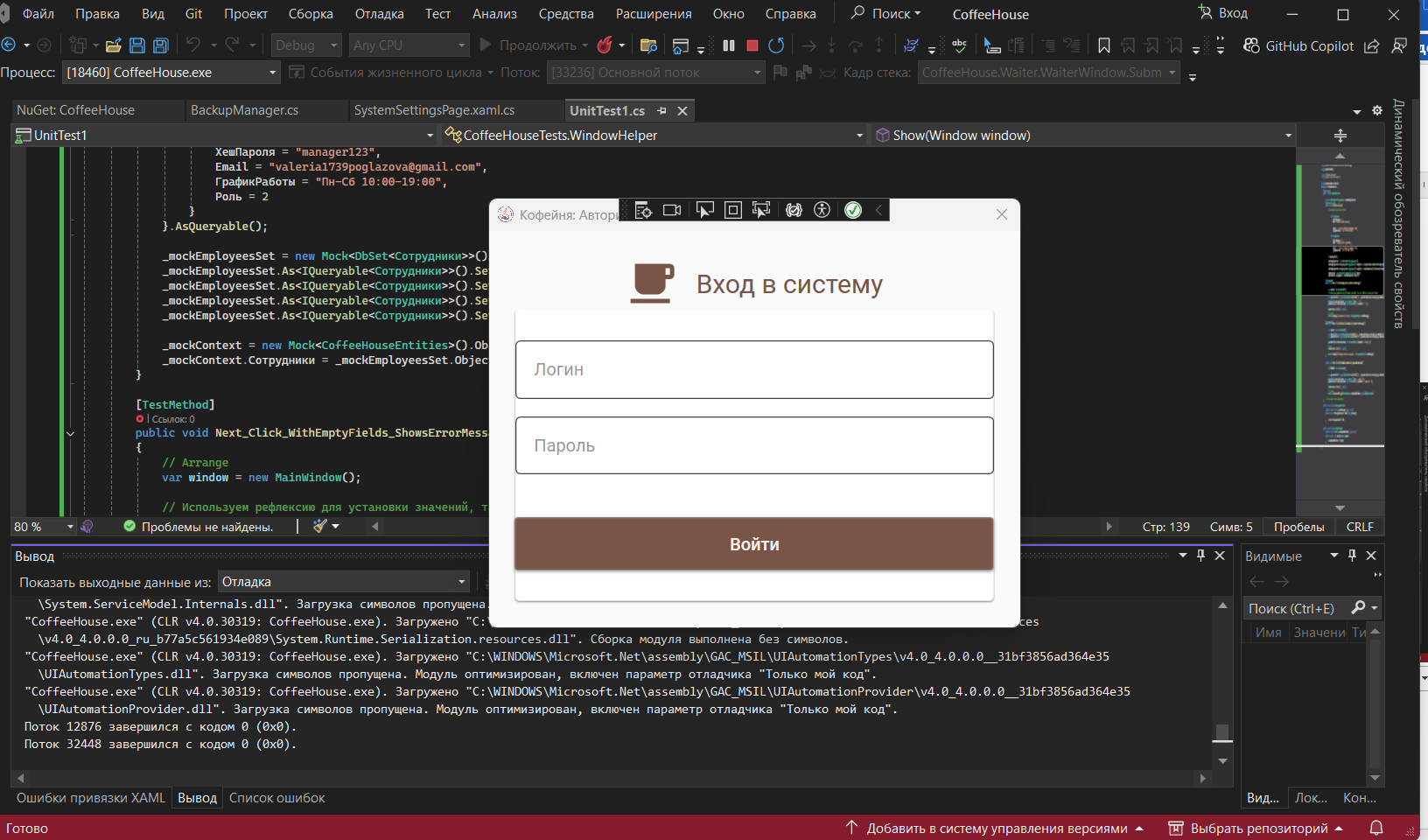


Рисунок 5.1 – Отладка настольного приложения

Отладка библиотеки классов (Рисунок 5.2):

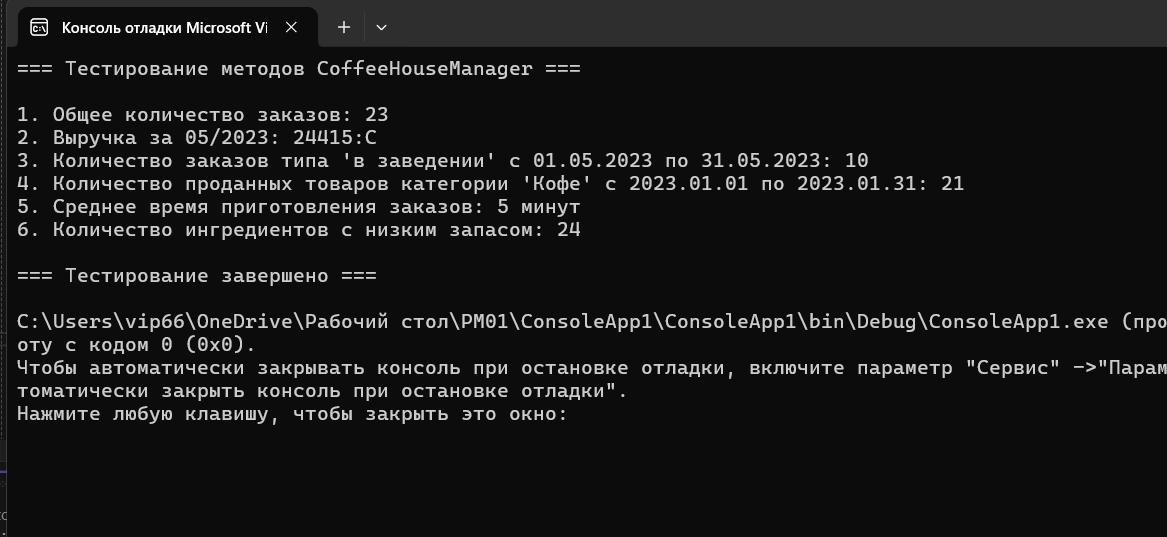


Рисунок 5.2 – Отладка библиотеки

Отладка api приложения (Рисунок 5.3):

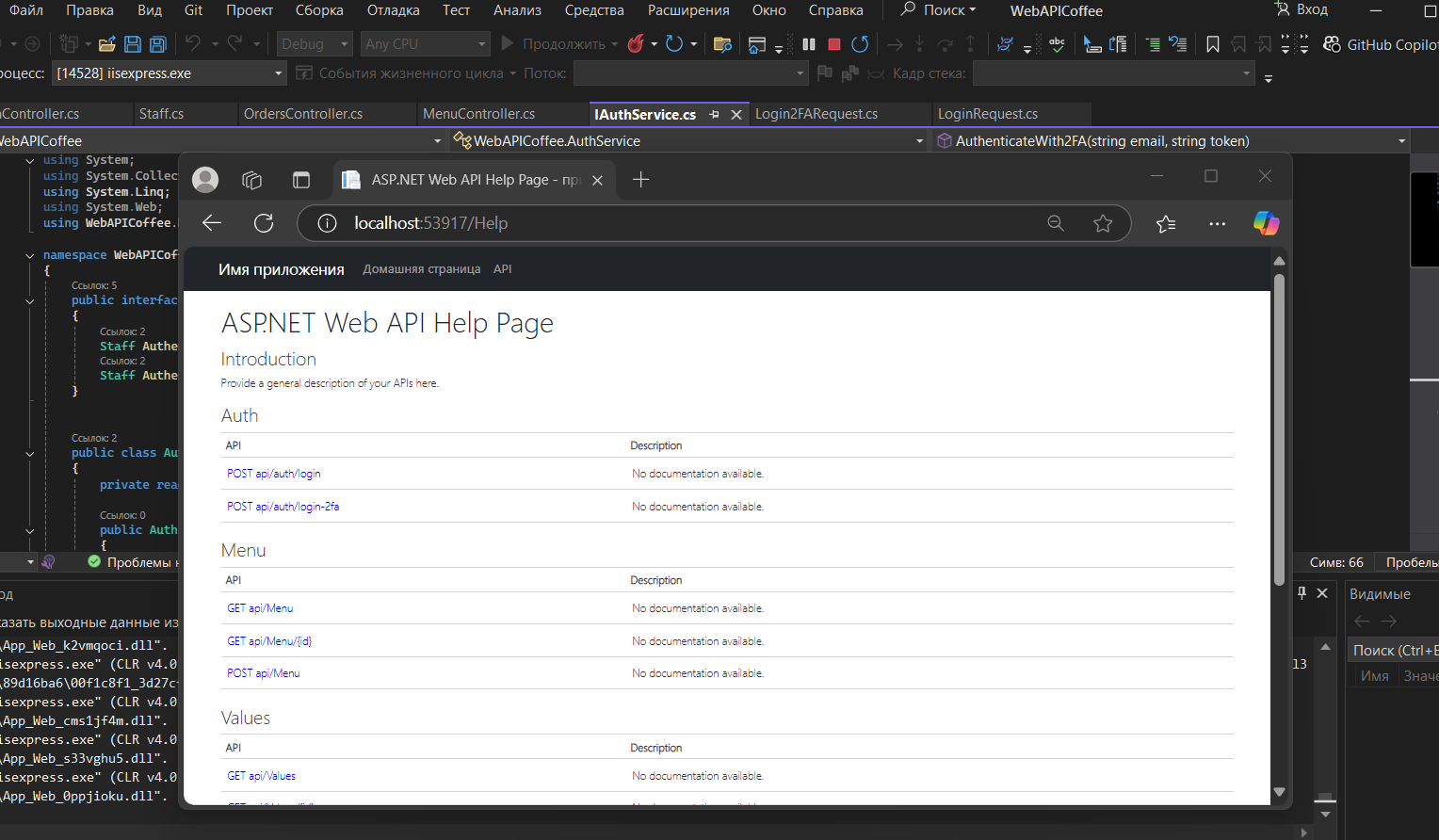


Рисунок 5.3 – Отладка api приложения

Отладка программного модуля позволила выявить и устранить ошибки, а также оптимизировать работу системы.

# 6. ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ

Тестирование является ключевым этапом, который позволяет убедиться в корректности работы системы и ее соответствии техническому заданию. В этом разделе описаны виды тестирования, проведенные для программного модуля.

Проведено тестирование:

1. Интеграционное тестирование библиотеки.

• Для тестирования библиотеки классов было создано консольное приложение. Это приложение позволяло вызывать методы библиотеки и проверять их функциональность в реальном времени. Консольное приложение обеспечивало удобный интерфейс для тестирования различных сценариев и проверки корректности работы библиотечных функций.

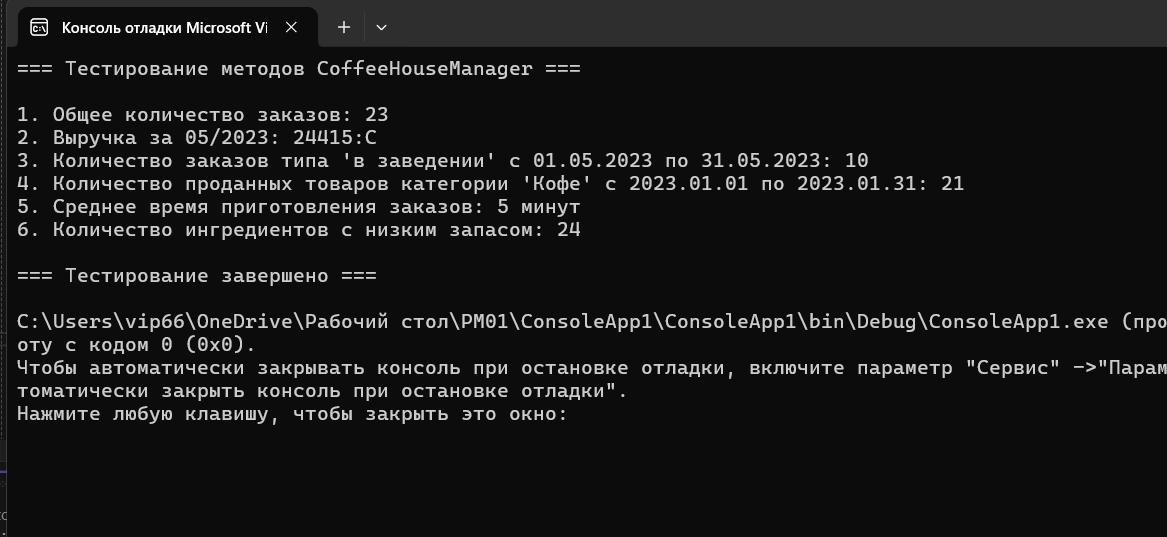


Рисунок 6.1 - Тестирование библиотеки

1. Модульное тестирование настольного приложения   
   Для тестирования настольного приложения был создан проект модульного теста (.NET Framework). Юнит-тестирование позволило выявить ошибки на ранних этапах разработки и убедиться в том, что каждая часть приложения работает корректно и в соответствии с требованиями.

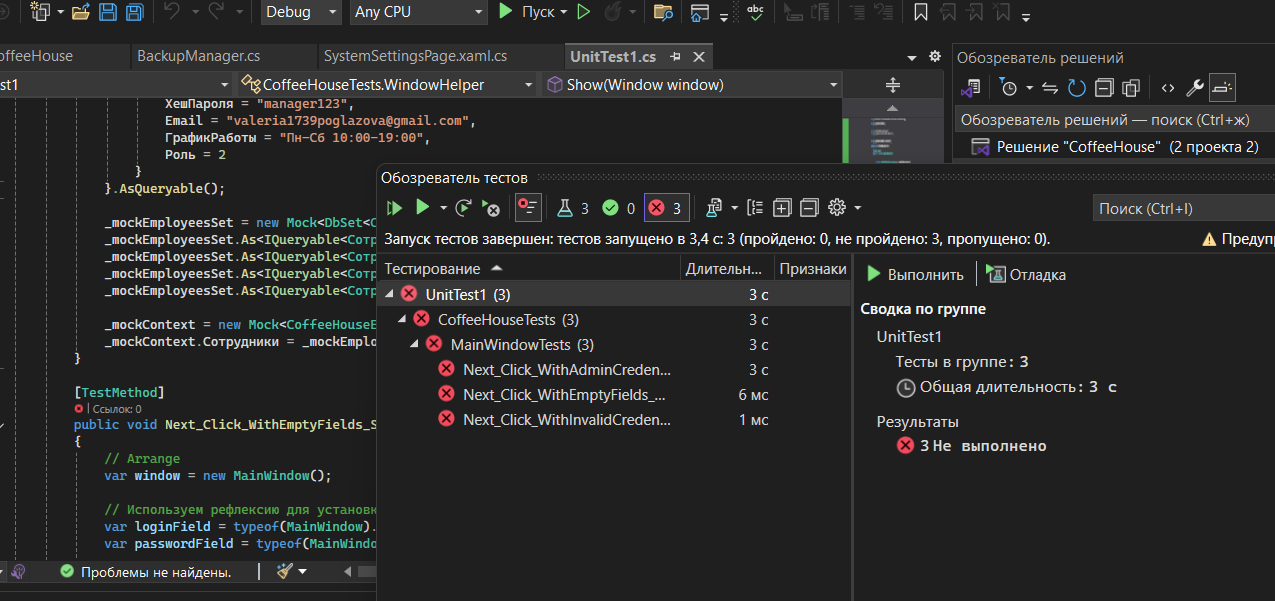


Рисунок 6.1 - Тестирование настольного приложения

1. Нагрузочное тестирование (API).

Нагрузочное тестирование было проведено с целью оценки производительности и стабильности API при обработке множества запросов. Тестирование включало в себя отправку запросов типа GET и POST с использованием программы Postman. Однако, в процессе тестирования было выявлено, что не все методы API работают корректно. Некоторые конечные точки возвращали неожиданные результаты или ошибки, что требует дополнительного внимания и исправления.

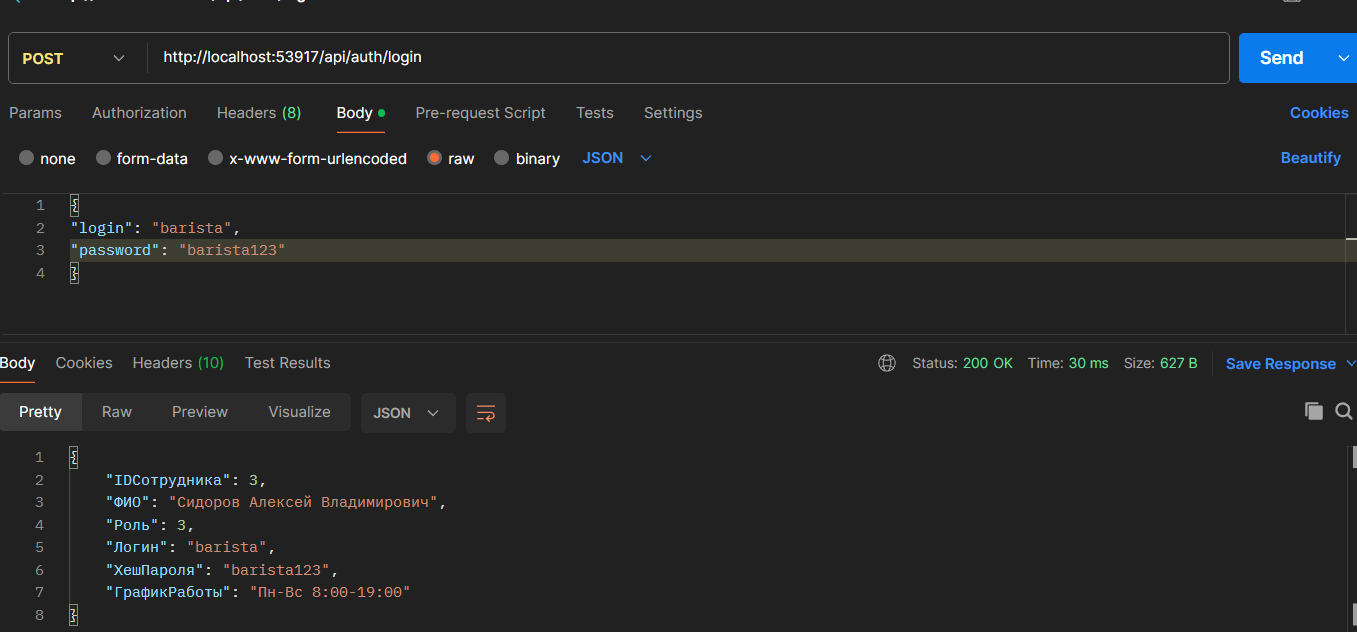


Рисунок 6.3 – Запрос в Postman (авторизация пользователя)

Использование различных методов тестирования для каждого компонента системы обеспечило высокое качество программного обеспечения и минимизировало количество ошибок. Тем не менее, в процессе тестирования были выявлены проблемы с некоторыми методами API, которые требуют дополнительного внимания. Все тесты, которые были успешно выполнены, подтверждают готовность системы к развертыванию и использованию, но необходимо устранить выявленные недостатки перед окончательным запуском.

# 7. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

Оформление результатов тестирования является важным этапом, который позволяет систематизировать полученные данные, выявить ошибки и определить направления для улучшения системы. В этом разделе представлены результаты тестирования, включая описание тестовых случаев, выявленные проблемы и рекомендации по их устранению.

1. Описание тестовых случаев:

В процессе тестирования были разработаны и выполнены 10 тестовых случаев, охватывающих ключевые функциональные и нефункциональные требования системы. Каждый тестовый случай включал:

* Название теста.
* Приоритет теста (высокий, средний).
* Ожидаемый результат.
* Фактический результат.
* Статус выполнения (Pass/Fail).
* Комментарии и рекомендации.

1. Результаты выполнения тестов:

Все результаты тестирования, включая отчеты, были загружены в репозиторий на GitHub. Это обеспечивает удобное хранение, версионность и доступ к документам для всех участников проекта. Каждый отчет был размещен в папке репозитория, что позволяет легко находить нужные материалы и отслеживать изменения.

Таким образом, результаты тестирования были тщательно проанализированы и оформлены в удобном для использования виде, что позволяет эффективно работать с ними на всех этапах жизненного цикла проекта.

# 8. РЕФАКТОРИНГ И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА

Рефакторинг и оптимизация кода — это важные этапы, которые позволяют улучшить читаемость, производительность и поддерживаемость программного модуля. В данном разделе описаны основные изменения, внесенные в код.

В процессе разработки программного модуля были проведены следующие работы по рефакторингу и оптимизации:

1. Использование блоков try-catch:

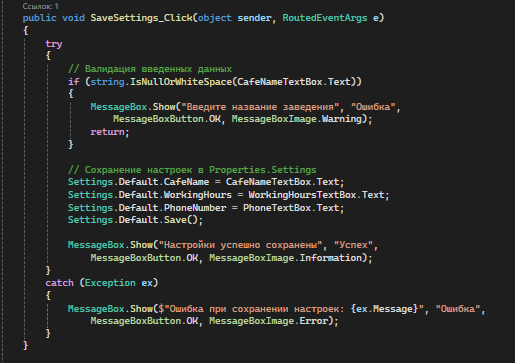


Рисунок 8.1 – Рефакторинг и оптимизация кода (блоки try-catch)

1. Проверка на пустые значения:

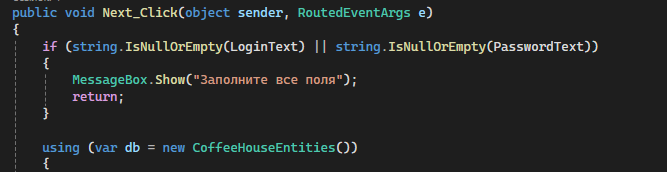


Рисунок 8.2 – Рефакторинг и оптимизация кода (пустые значения)

1. Именование переменных по стилю CamelCase:

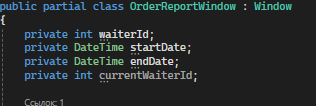


Рисунок 8.3 – Рефакторинг и оптимизация кода (CamelCase)

1. Структуризация кода с использованием циклов и методов:

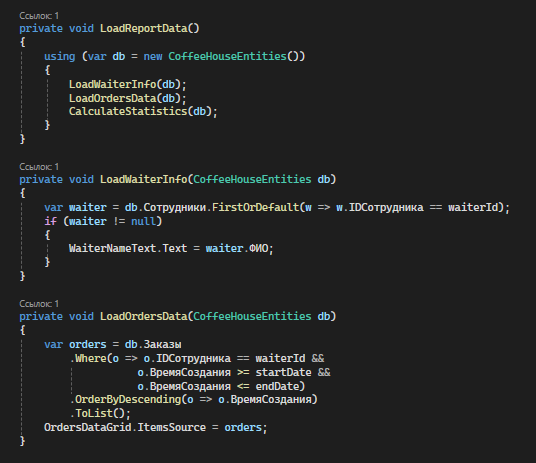


Рисунок 8.4 – Рефакторинг и оптимизация кода (циклы и методы)

1. Добавление комментариев:

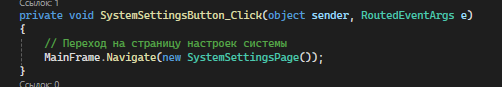


Рисунок 8.4 – Рефакторинг и оптимизация кода (комментирование)

Рефакторинг и оптимизация кода позволили значительно улучшить качество программного модуля. Код стал более структурированным, производительным и удобным для дальнейшей разработки и поддержки. Все изменения были зафиксированы в системе контроля версий, что обеспечивает прозрачность и возможность отката в случае необходимости.

# 9. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА

Мобильное приложение является важной частью системы, обеспечивающей удобство работы для пользователей, особенно для официанта. В этом разделе описаны этапы разработки мобильного приложения.

**Технологии разработки мобильного приложения:**

Платформа:

• Visual Studio 2022 - Xamarin.Forms (Android)

База данных:

• SQLite (локальная)

Библиотеки для работы с базой данных:

• SQLite-net-pcl,

• SQLiteNetExtensions

**Особенности реализации:**

Пользовательский интерфейс:

* Интуитивно понятный и отзывчивый интерфейс
* Адаптивный дизайн для различных размеров экранов
* Удобная навигация между разделами

Работа с данными:

* Полноценная система CRUD-операций
* Сложные запросы для получения связанных данных
* Автоматическое заполнение тестовыми данными при первом запуске

Безопасность:

* Хранение учетных данных в защищенном виде
* Разграничение прав доступа

Производительность:

* Оптимизированные запросы к базе данных
* Минимальное потребление ресурсов устройства

1. Функция авторизации.

Позволяет авторизованным пользователям получить доступ к функциям приложения. Требует ввода логина и пароля.

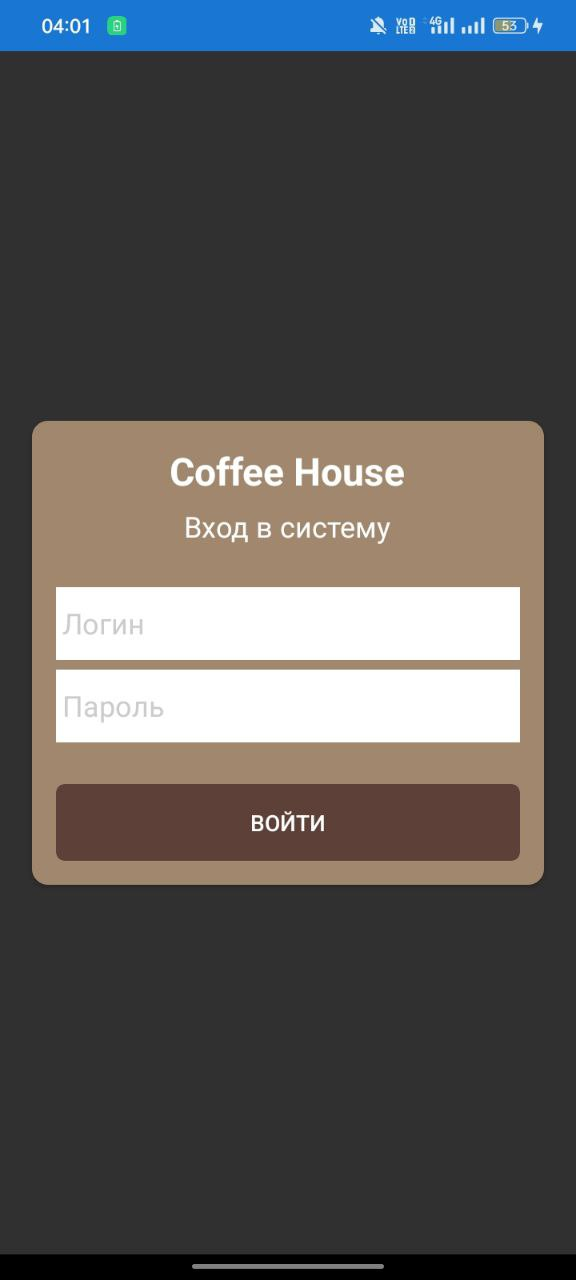


Рисунок 9.1. – Окно авторизации

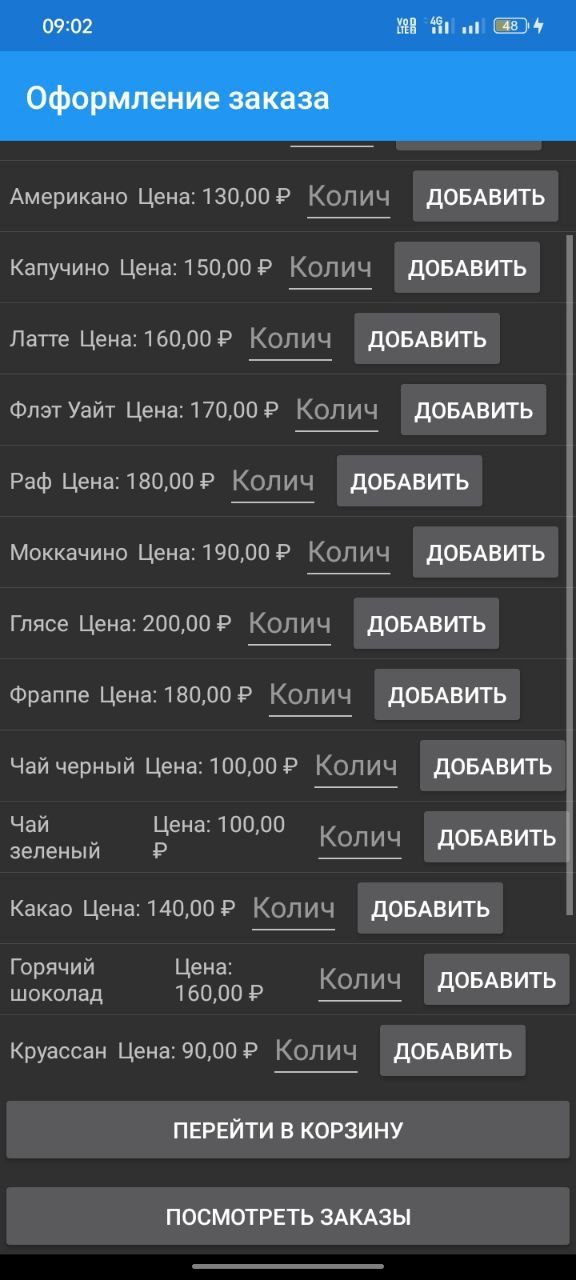


Рисунок 9.2. – Окно оформления заказа



Рисунок 9.3. – Окно корзины заказа



Рисунок 9.4. – Окно текущих заказов

**Тестирование мобильного приложения**

В процессе тестирования мобильного приложения были разработаны и выполнены тест-кейсы, охватывающие основные функциональные сценарии. Все тест-кейсы загружены в репозиторий GitHub для обеспечения прозрачности процесса тестирования и дальнейшего аудита.

В ходе проверки выявлены следующие критические ошибки:

* Отмена записи не работает
* Функция записи к врачу не работает

**Тестирование проводилось на:**

* Эмуляторах Android (версии 10, 11, 12)
* Физических устройствах (TCL 40R 5G)

# 

# 10. ЗАКЛЮЧЕНИЕ

Производственная практика ПМ01. «Разработка модулей программного обеспечения для компьютерных систем» проходила с 20 марта по 21 апреля 2025 г. в организации ООО «Электрон-Софт».

В ходе практики я получила ценный опыт в области разработки программных модулей, который позволил мне закрепить теоретические знания и приобрести практические навыки, необходимые для работы в данной сфере.

Производственная практика позволила закрепить полученные теоретические знания на учебной практике и приобрести необходимые навыки в области осуществлении разработки программных модулей.

Полученные знания и опыт будут использованы в дальнейшей профессиональной деятельности.

# ПРИЛОЖЕНИЯ

Все материалы, программные модули и документация, созданные в рамках проекта, были систематизированы и загружены в репозиторий на GitHub.

Ссылка на репозиторий: <https://github.com/Valeria12-10/PrPractikPM01>