МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по ученой практике**

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

**Тема: «Разработка программного модуля «Складской учет»»**

Студент

Поглазова Валерия Владимировна

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

подпись

2024-2025 уч. год

**Содержание**

1. Сведения об организации (базе практики)
2. Разработка и анализ технического задания
3. Разработка программного модуля по алгоритму в соответствии техническому заданию
4. Оформление документации на программный модуль
5. Выполнение отладки программного модуля
6. Выполнение тестирования программного модуля
7. Оформление результатов тестирования
8. Рефакторинг и оптимизации программного кода
9. Разработка программного модуля для мобильного устройства
10. Проверка работоспособности программного модуля на мобильном устройстве или эмуляторе
11. Заключение.
12. Приложения к отчету: диск, отчет в электронном виде, презентация для выступления, проект программного модуля, проект мобильного приложения и др. материалы.

# Сведения об организации (базе практики)

Наименование базы практики – Кировское областное государственное профессиональное образовательное бюджетное учреждение «Слободской колледж педагогики и социальных отношений» (КОГПОБУ СКПиСО).

Адрес – Кировская область, г. Слободской, ул. Рождественская, д. 69.

ФИО руководителя (директора) – Шеренцова Ольга Михайловна.

Телефон директора +79014791707.

Образование в КОГПОБУ СКПиСО ориентировано на массовое обучение специалистов с рабочими навыками. Половина учебного времени отводится на практические работы в лабораториях. Это помогает выпускникам органично и быстро включаться в трудовые процессы на новых местах работы.

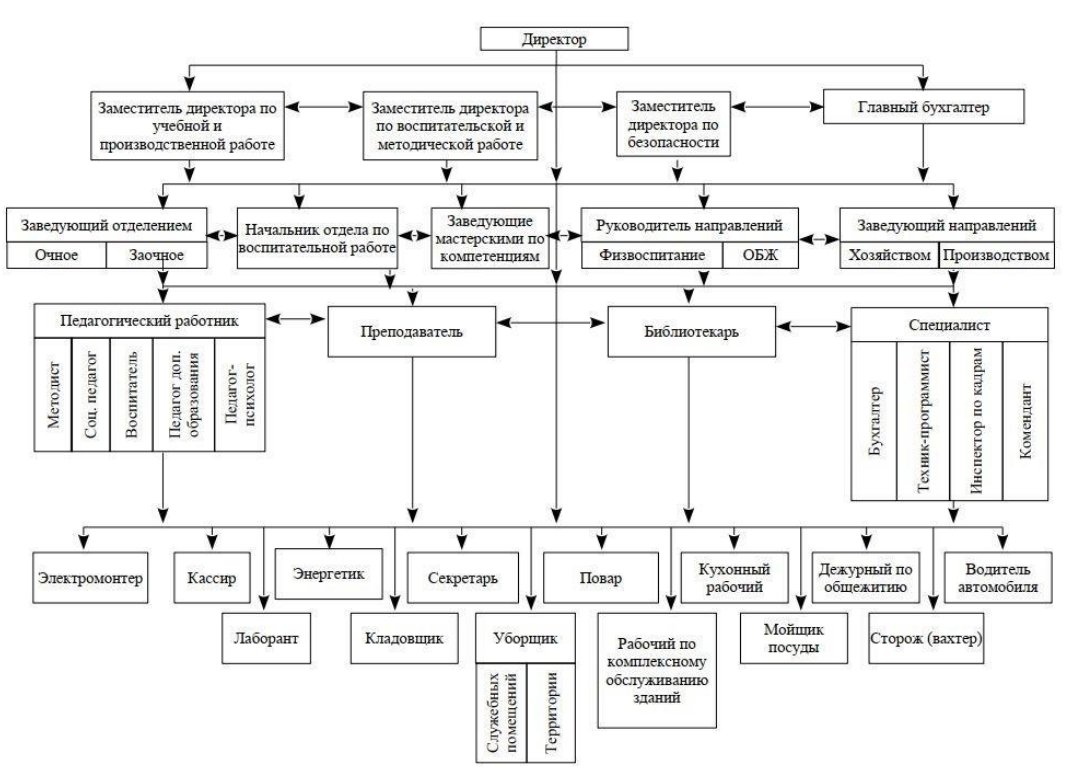
Схема организационной структуры представлена на рисунке 1.

Рисунок 1 – Схема организационной структуры

# 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, Кладовщика, Менеджера по продажам и Бухгалтера) с системой (Рисунок 2).



Рисунок 2 – Диаграмма вариантов использования

* Диаграмма последовательности (Sequence Diagram) показывает последовательность формирования заказа, сборку заказа и передача пользователю товаров (Рисунок 3).



Рисунок 3 – Диаграмма последовательности

* Диаграмма видов деятельности (Activity Diagram) (Рисунок 4).

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

* ER-диаграмма (Entity-Relationship Diagram).

Диаграмма отображает структуру базы данных, включая сущности, их атрибуты и связи между ними (Рисунок 5).

1. На основе ER-диаграммы была спроектирована база данных, реализованная в MS SQL Server (Рисунок 6).

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 4 – Диаграмма видов деятельности



Рисунок 5 – ER-диаграмма

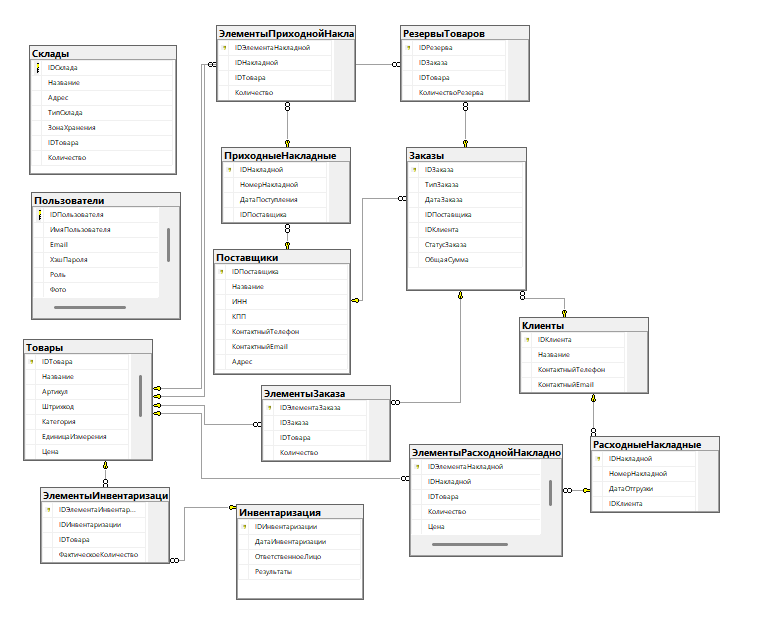


Рисунок 6 – Диаграмма базы данных

# Разработка программного модуля по алгоритму в соответствии техническому заданию

Разработка программного модуля началась с анализа технического задания и определения ключевых функциональных требований. Основной задачей стало создание API, библиотеки для работы с данными и настольного приложения, которое обеспечит удобное взаимодействие пользователей с системой складского учета.

API разработано с необходимым функционалом по техническому заданию. Реализованы методы GET и POST запросов (Рисунок 7):

1. Авторизация (2 способа).
2. Получение списка товаров на складе.
3. Получение списка складов.
4. Отправка данных о товаре.



Рисунок 7 – Разработка API

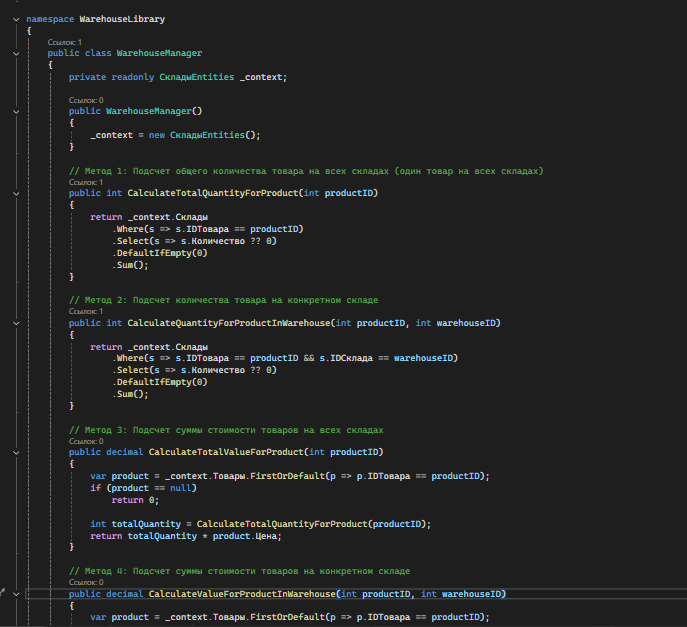
Создана библиотека для получения информации со складов (Рисунок 8):

1. Подсчет количества товаров (общее) по складам (один товар на всех складах) и на складе (сумма позиций).
2. Подсчет суммы стоимости товаров на складе и по складам.
3. Подсчет товара по категориям на складе и по складам.



Рисунка 8 – Библиотека

На рисунке 9 изображен небольшой код библиотеки.



Рисунка 9 – Код библиотеки

Создание настольного приложения:

1. Реализована двухфакторная аутентификация и разграничение по ролям:

Администратор имеет возможность работать со складами, товарами, клиентами (Рисунок 10), поставщиками (Рисунок 11) и пользователями системы (редактирование, удаление, добавление). Также реализована функция поиска нужной информации на каждой вкладке.

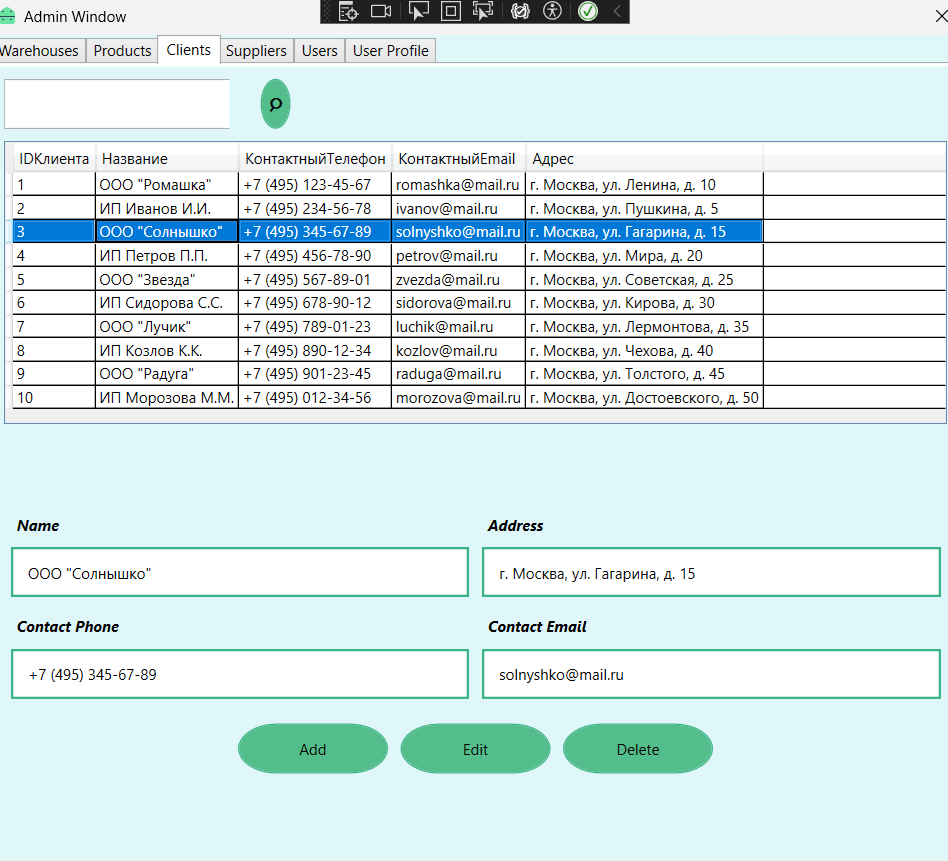


Рисунок 10 – Окно администратора

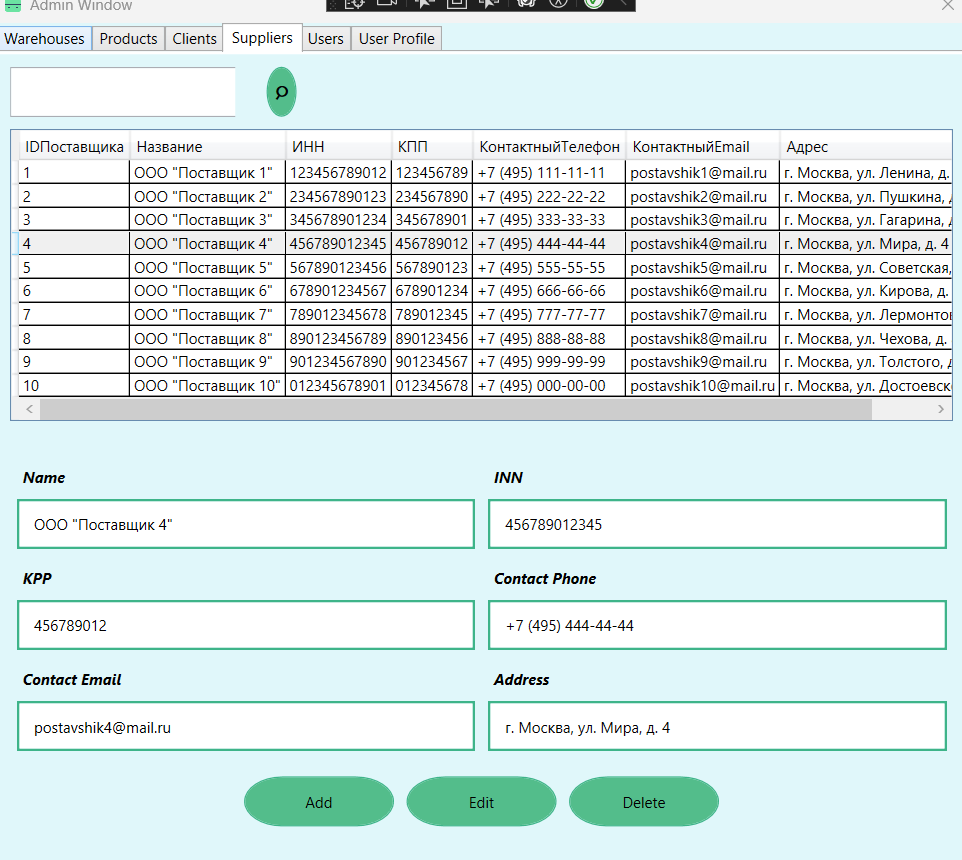


Рисунок 11 – Окно администратора

Кладовщик имеет возможность работать с накладными (расходными, приходными), формировать отчет по инвентаризации (экспорт в Excel), формирование штрих-кода для товара и вывод его на печать (Рисунок 12). Также реализован профиль пользователя.

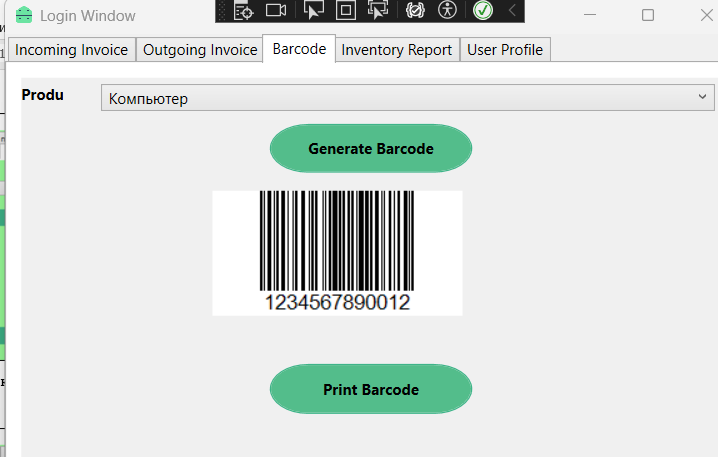


Рисунок 12 – Окно кладовщика

Менеджер по продажам имеет возможность формировать заказы и контролировать их (клиенту, поставщику), просматривать и контролировать склады (Рисунок 13), накладные (Рисунок 14), товары, регистрировать клиентов и поставщиков. Также реализован профиль пользователя возможность поиска информации.

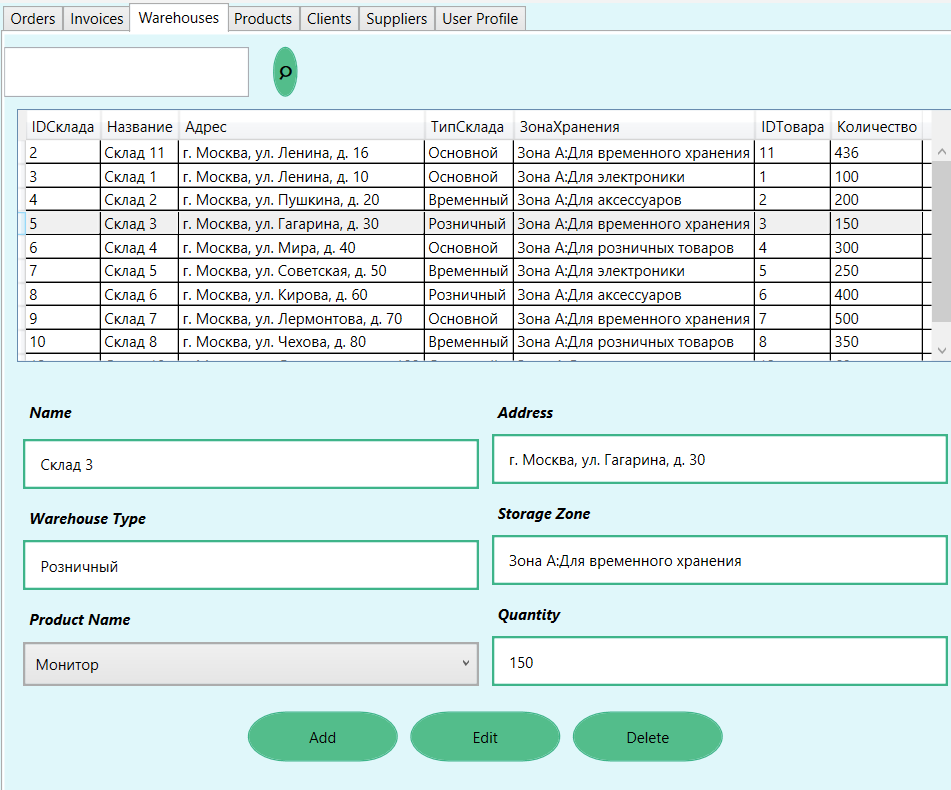


Рисунок 13 – Окно менеджера по продажам

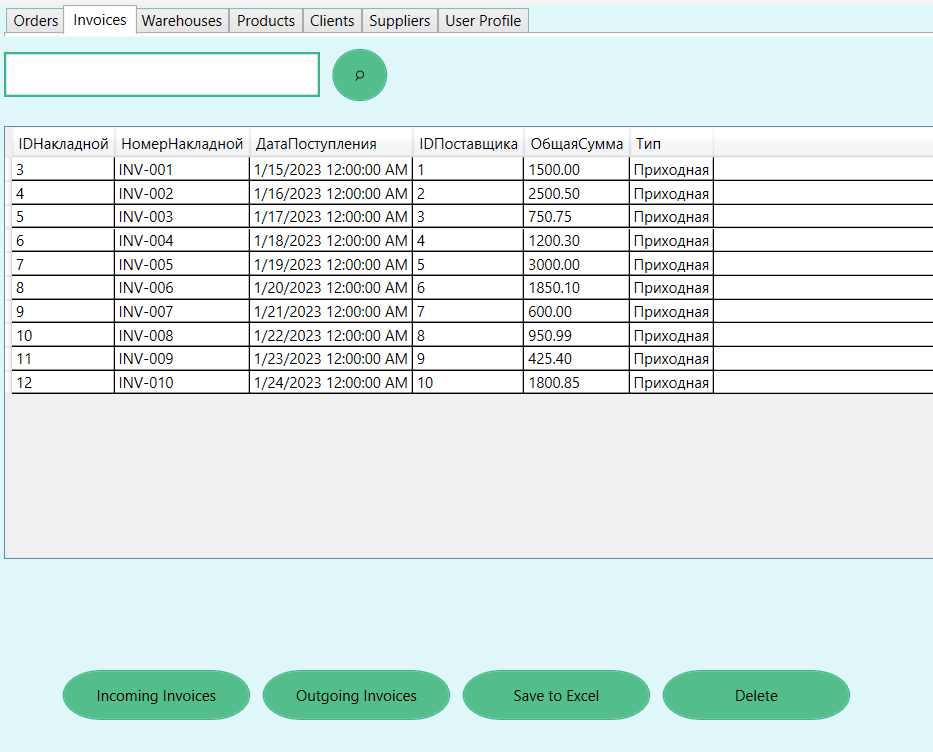


Рисунок 14 – Окно менеджера по продажам

Бухгалтер имеет возможность формировать отчеты по инвентаризации (проверка фактических и ожидаемых значений склада, вывод расхождений), по складам (сумма, количество, категория), по оборотам, по остаткам. Также реализован профиль пользователя. (Рисунок 15).

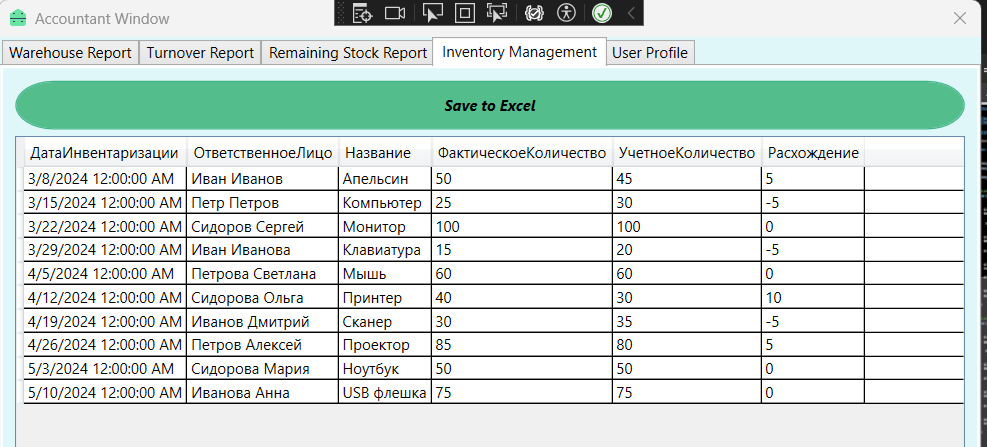


Рисунок 15 – Окно бухгалтера

1. Реализованы функции поиска и сортировки данных (заказы, накладные, товары).
2. Реализовано окно профиля пользователя с возможностью редактирования Фамилии, Имени, Отчества, Логина, есть возможность установить новый пароль и сменить фото пользователя.

На рисунке 16 изображено окно профиля для пользователя, вошедшего под ролью администратора.

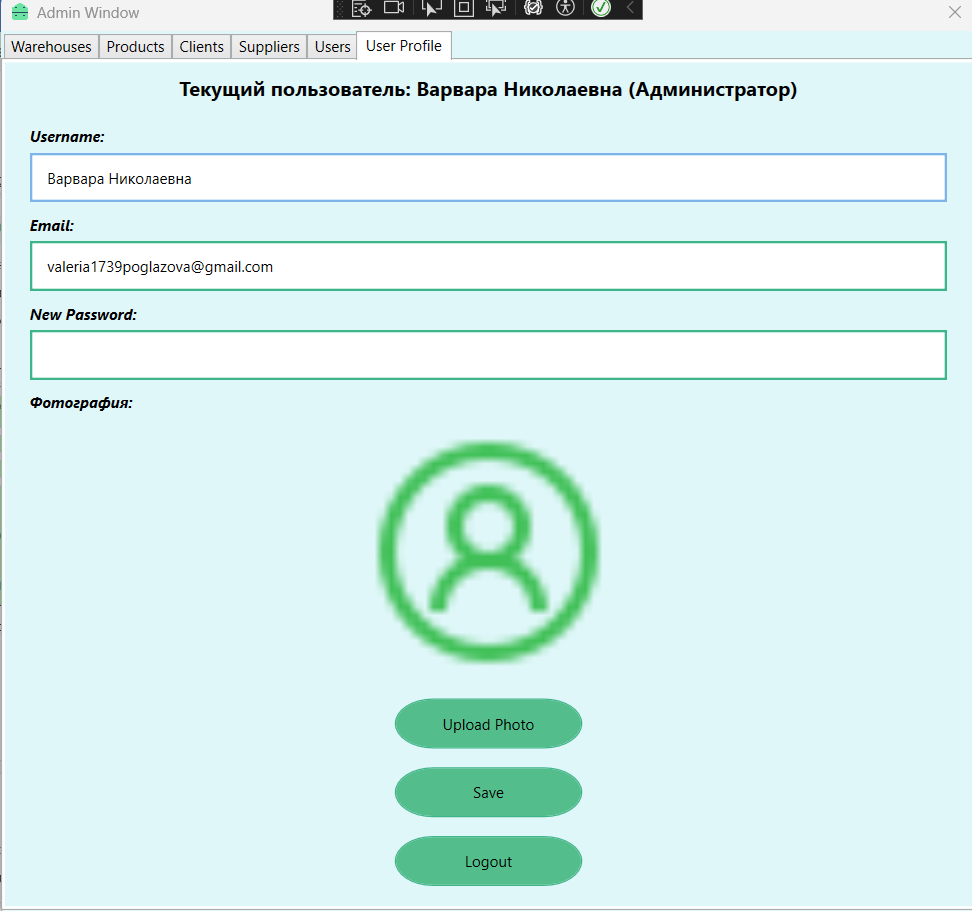


Рисунок 16 – Окно профиля пользователя

1. Изображения и файлы хранятся на стороне БД.

В таблице «Пользователь» для фото пользователя добавлено поле с соответствующим типом данных.

1. Навигация по приложению реализована с помощью Mainframe (Page).

На рисунке 17 изображена часть структуры настольного приложения.

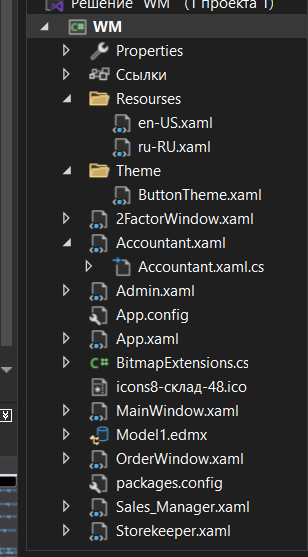


Рисунок 17 – Структура приложения

Создание мобильного приложения:

1. Мобильное приложение предназначено для роли кладовщика, но просмотр данных могут осуществлять и другие роли.
2. Функции мобильного приложения:

* Двухфакторная аутентификация.

На рисунке 18 изображена функция авторизации пользователя.

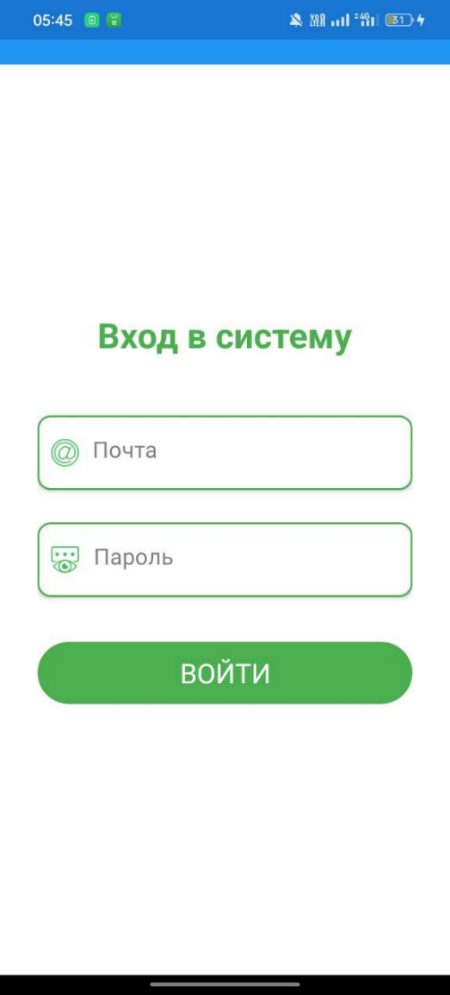


Рисунок 18 – Мобильное приложение

* Меню со следующими фрагментами:

Список складов (рисунок 19) и их местоположение.

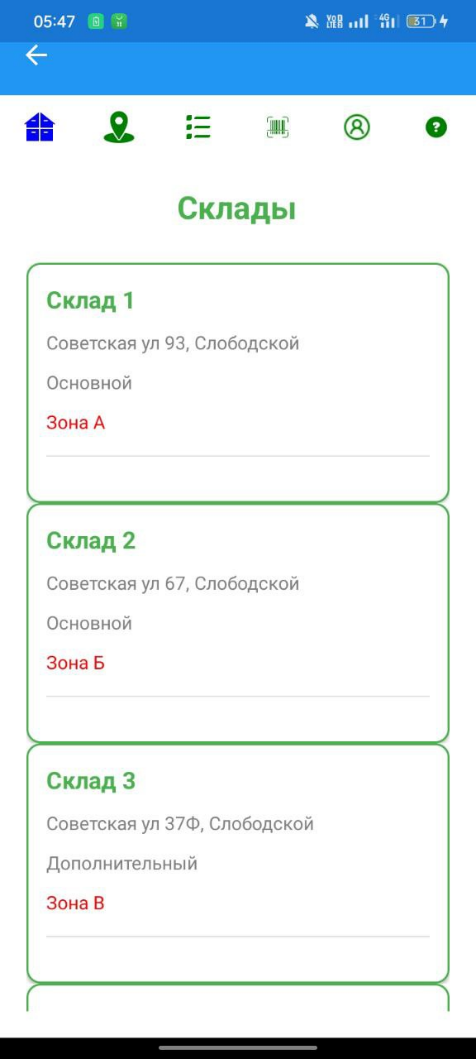


Рисунок 19 - Мобильное приложение

* Список товаров по складам.
* Сканирование штрихкода для проверки товара (Рисунок 20).

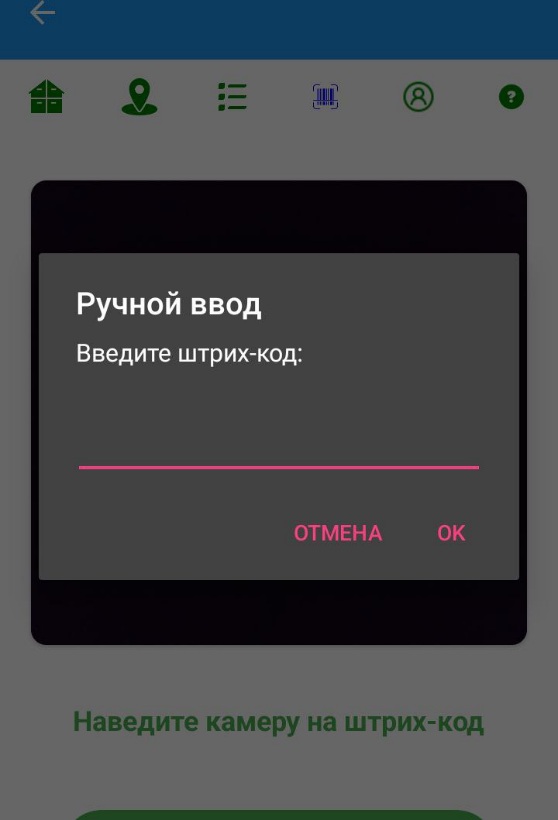
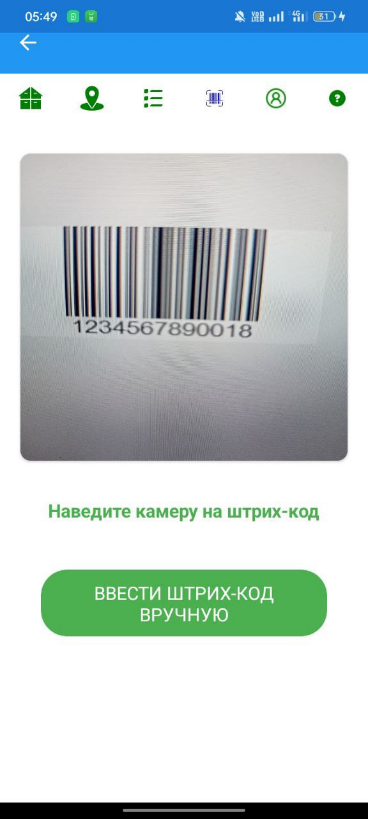


Рисунок 20 - Мобильное приложение

1. Справка о приложении (рисунок 21).

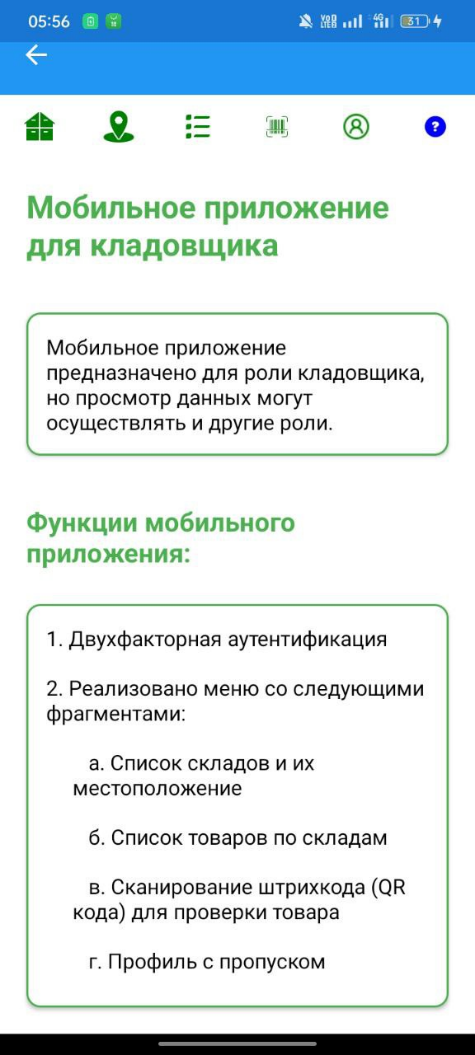


Рисунок 21 - Мобильное приложение

1. Профиль с пропуском (рисунок 22).

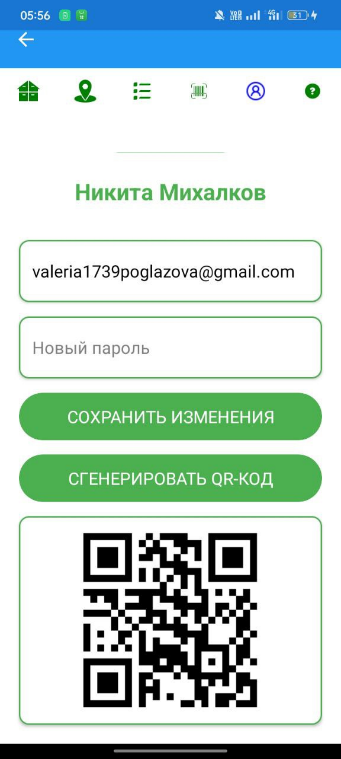
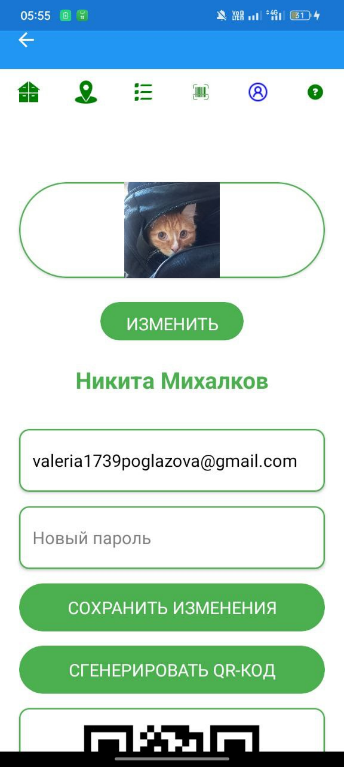


Рисунок 22 - Мобильное приложение

1. Реализована возможность сохранения данных авторизации.

# Оформление документации на программный модуль

Качественная документация является важной частью любого программного продукта. Она обеспечивает удобство использования, поддержки и дальнейшего развития системы. В этом разделе описаны этапы создания документации для программного модуля.

Разработано руководство оператора для мобильного приложения (Рисунок 23).

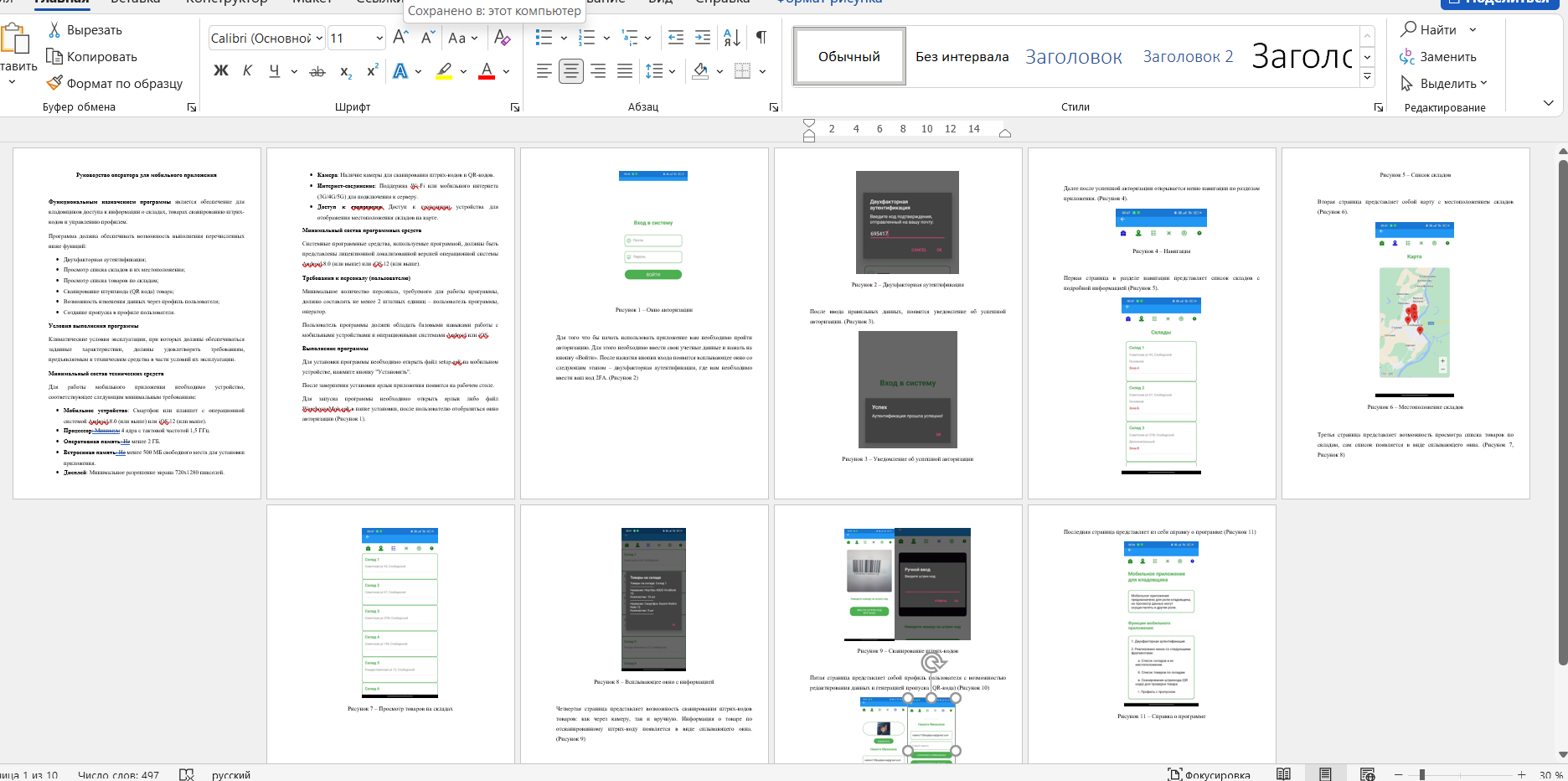


Рисунок 23 – Руководство оператора для мобильного приложения

Разработано руководство программиста для API (Рисунок 24).

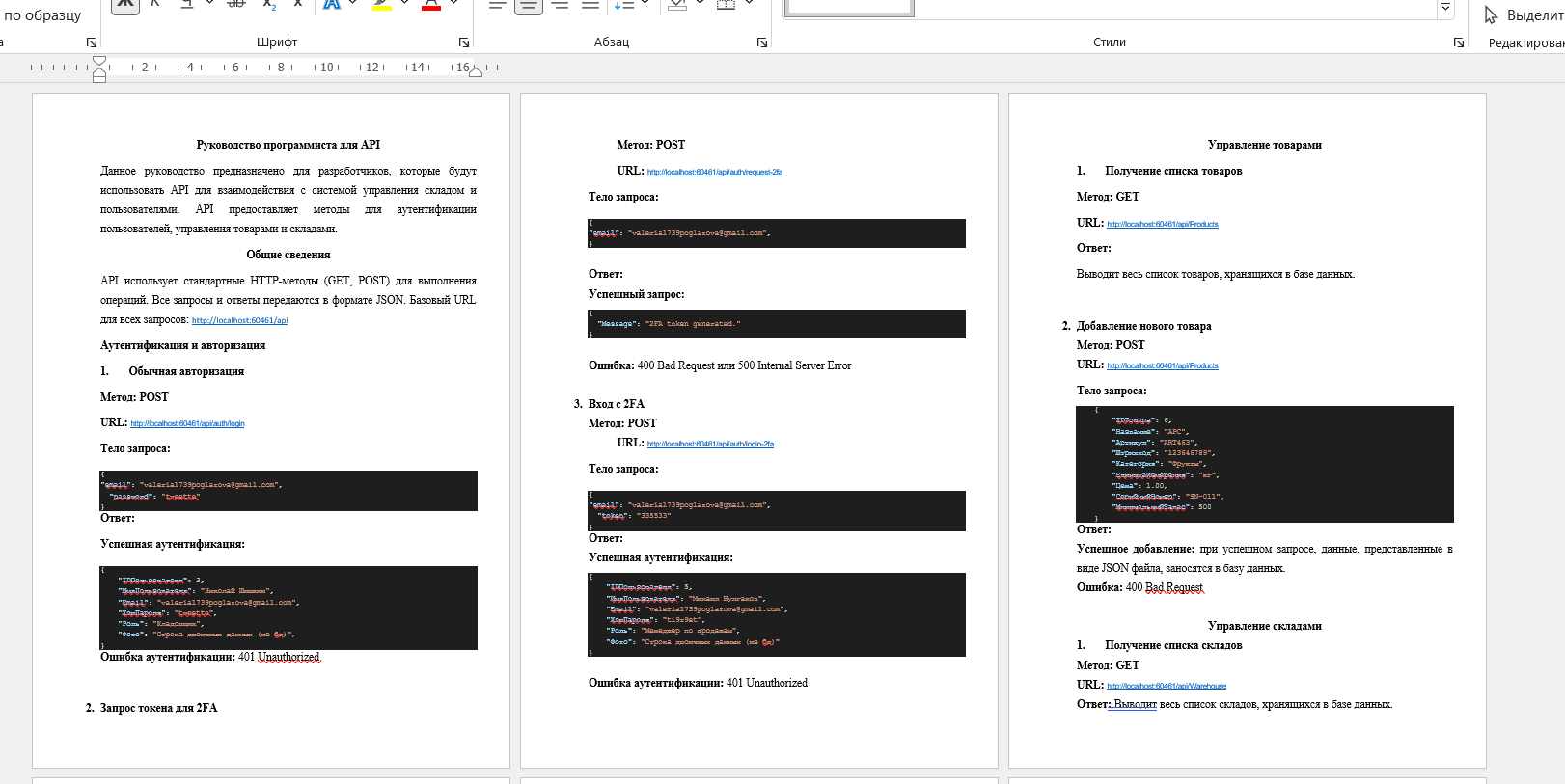


Рисунок 24 – Руководство программиста для API

Добавлена справка о программе с описанием функций в мобильное приложение (Рисунок 25).

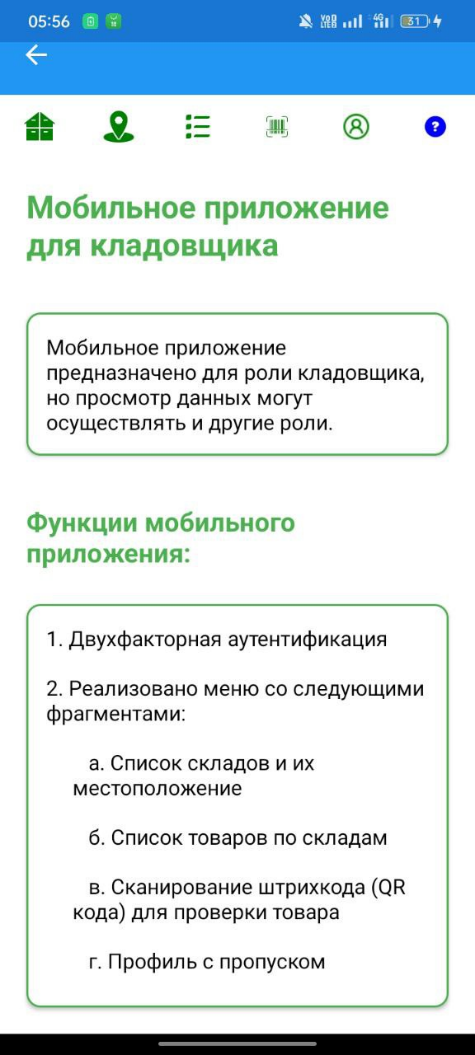


Рисунок 25 – Справка о программе

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

# 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.

Отладка программного модуля проводилась на всех этапах разработки, начиная с тестирования отдельных компонентов и заканчивая интеграцией всех частей системы. Основные этапы отладки включали:

1. Исправление ошибок в коде:

Использование инструментов отладки в среде разработки Visual Studio для поиска и устранения синтаксических и логических ошибок.

Проверка корректности работы методов API, включая обработку исключений и валидацию входных данных.

Устранение ошибок, связанных с работой базы данных, таких как некорректные запросы SQL или проблемы с подключением.

1. Оптимизация работы API:

Проведен анализ производительности API с использованием программы Postman.

Устранены узкие места, такие как медленные запросы к базе данных и избыточность.

1. Проверка корректности работы всех функций приложения:

Проверена корректность работы всех ролей пользователей (администратор, кладовщик, менеджер по продажам, бухгалтер).

Убедились, что все функции, такие как добавление товаров, оформление накладных, сканирование штрихкодов и формирование отчетов, работают без ошибок.

Проверена корректность работы мобильного приложения в режиме оффлайн.

1. Тестирование пользовательского интерфейса:

Проверена корректность отображения всех элементов интерфейса на разных разрешениях экрана.

Убедились, что навигация между экранами происходит без задержек и ошибок.

Проверена работа всех кнопок, форм и других интерактивных элементов.

1. Интеграционная отладка:

Проверена корректность взаимодействия между всеми компонентами системы: API, базой данных, настольным и мобильным приложениями.

Результаты отладки:

1. Все выявленные ошибки были успешно устранены.
2. Производительность системы была значительно улучшена.
3. Пользовательский интерфейс стал более стабильным и удобным в использовании.
4. Система готова к тестированию и дальнейшему внедрению.

Отладка программного модуля позволила выявить и устранить ошибки, а также оптимизировать работу системы. В результате была достигнута стабильность и высокая производительность, что подтверждает готовность системы к использованию в реальных условиях.

.

# 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 был создан проект модульного теста (.NET Framework).

Результаты изображены на рисунке 26.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WarehouseLibrary;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

// Создаем экземпляр WarehouseManager

var warehouseManager = new WarehouseManager();

// Тест 1: Подсчет общего количества товара на всех складах

int productID = 1; // Пример ID товара

int totalQuantity = warehouseManager.CalculateTotalQuantityForProduct(productID);

Console.WriteLine($"Общее количество товара с ID {productID} на всех складах: {totalQuantity}");

// Тест 2: Подсчет количества товара на конкретном складе

int warehouseID = 1; // Пример ID склада

int quantityInWarehouse = warehouseManager.CalculateQuantityForProductInWarehouse(productID, warehouseID);

Console.WriteLine($"Количество товара с ID {productID} на складе {warehouseID}: {quantityInWarehouse}");

// Тест 3: Подсчет суммы стоимости товаров на всех складах

decimal totalValue = warehouseManager.CalculateTotalValueForProduct(productID);

Console.WriteLine($"Общая стоимость товара с ID {productID} на всех складах: {totalValue}");

// Тест 4: Подсчет суммы стоимости товаров на конкретном складе

decimal valueInWarehouse = warehouseManager.CalculateValueForProductInWarehouse(productID, warehouseID);

Console.WriteLine($"Стоимость товара с ID {productID} на складе {warehouseID}: {valueInWarehouse}");

// Тест 5: Подсчет количества товаров по категориям на всех складах

Dictionary<string, int> quantityByCategory = warehouseManager.CalculateQuantityByCategory();

Console.WriteLine("Количество товаров по категориям на всех складах:");

foreach (var category in quantityByCategory)

{

Console.WriteLine($"{category.Key}: {category.Value}");

}

// Тест 6: Подсчет количества товаров по категориям на конкретном складе

Dictionary<string, int> quantityByCategoryInWarehouse = warehouseManager.CalculateQuantityByCategoryInWarehouse(warehouseID);

Console.WriteLine($"Количество товаров по категориям на складе {warehouseID}:");

foreach (var category in quantityByCategoryInWarehouse)

{

Console.WriteLine($"{category.Key}: {category.Value}");

}

// Отрицательные тесты

// Тест 7: Подсчет общего количества для несуществующего товара

int nonExistentProductID = 999;

int totalQuantityNonExistent = warehouseManager.CalculateTotalQuantityForProduct(nonExistentProductID);

Console.WriteLine($"Общее количество несуществующего товара с ID {nonExistentProductID}: {totalQuantityNonExistent}");

// Тест 8: Подсчет количества для несуществующего товара на складе

int quantityNonExistentInWarehouse = warehouseManager.CalculateQuantityForProductInWarehouse(nonExistentProductID, warehouseID);

Console.WriteLine($"Количество несуществующего товара с ID {nonExistentProductID} на складе {warehouseID}: {quantityNonExistentInWarehouse}");

// Тест 9: Подсчет суммы стоимости для несуществующего товара

decimal totalValueNonExistent = warehouseManager.CalculateTotalValueForProduct(nonExistentProductID);

Console.WriteLine($"Общая стоимость несуществующего товара с ID {nonExistentProductID}: {totalValueNonExistent}");

// Тест 10: Подсчет суммы стоимости для несуществующего товара на складе

decimal valueNonExistentInWarehouse = warehouseManager.CalculateValueForProductInWarehouse(nonExistentProductID, warehouseID);

Console.WriteLine($"Стоимость несуществующего товара с ID {nonExistentProductID} на складе {warehouseID}: {valueNonExistentInWarehouse}");

}

}

}



Рисунок 26 - Результат выполнения тестов

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. В рамках тестирования была создана коллекция запросов, настроено количество повторений (100/1 000 итераций) для оценки поведения системы под нагрузкой.

Цели тестирования:

* Оценить производительность API при обработке множества запросов.
* Проверить стабильность работы системы под нагрузкой.
* Выявить возможные узкие места и ошибки при высокой нагрузке.
* Определить время отклика системы при различных типах запросов.

Инструменты и методы:

* Инструмент: Postman.
* Типы запросов: GET, POST.
* Количество повторений: 100/1 000 итераций для каждого запроса.
* Коллекция: В Postman была создана коллекция, содержащая тестируемые запросы.

Описание тестирования

1. Подготовка.

В программе Postman была создана коллекция, включающая запросы GET и POST.

Настроено количество повторений (100/1 000 итераций) для имитации нагрузки.

1. Выполнение тестирования.

Запущены запросы GET и POST с использованием функционала Postman для нагрузочного тестирования.

Каждый запрос был выполнен 100/1 000 раз подряд для оценки поведения системы под нагрузкой.

В процессе тестирования фиксировались следующие метрики:

* Время отклика (response time).
* Количество успешных и неудачных запросов.
* Коды ответов (HTTP status codes).

1. Результаты тестирования

Запросы POST для авторизации пользователя.

* Среднее время отклика: 894.
* Количество успешных запросов: 100.
* Количество неудачных запросов: 0.
* Коды ответов: 200 OK.

Запросы GET для вывода списка товаров на складе.

* Среднее время отклика: 3116.
* Количество успешных запросов: 100.
* Количество неудачных запросов: 0.
* Коды ответов: 200 OK.

Запросы GET для вывода списка складов.

* Среднее время отклика: 848.
* Количество успешных запросов: 100.
* Количество неудачных запросов: 0.
* Коды ответов: 200 OK.

1. Анализ результатов

Производительность:

Среднее время отклика запроса POST для авторизации пользователя составляет 894 мс, что является приемлемым для операции авторизации.

Среднее время отклика запроса GET для вывода списка товаров на складе составляет 3116 мс, что является достаточно высоким показателем. Это может быть связано с недостаточной оптимизацией запроса.

Среднее время отклика запроса GET для вывода списка складов составляет 848 мс, что является хорошим показателем для данного типа запроса.

Стабильность:

Все запросы (POST и GET) были выполнены успешно. Количество неудачных запросов равно 0, что свидетельствует о высокой стабильности системы при нагрузке в 100/1 000 итераций.

Коды ответов для всех запросов — 200 OK, что подтверждает корректность работы API.

Проблемы:

Основной проблемой является высокое время отклика для запросов GET, связанных с выводом списка товаров на складе (3116 мс). Это может негативно сказаться на пользовательском опыте, особенно при увеличении нагрузки.

Нагрузочное тестирование API показало, что система стабильно справляется с нагрузкой в 100/1 000 запросов для тестируемых запросов. Однако выявлено высокое время отклика для запросов GET, связанных с выводом списка товаров на складе.

Скриншоты тестирования

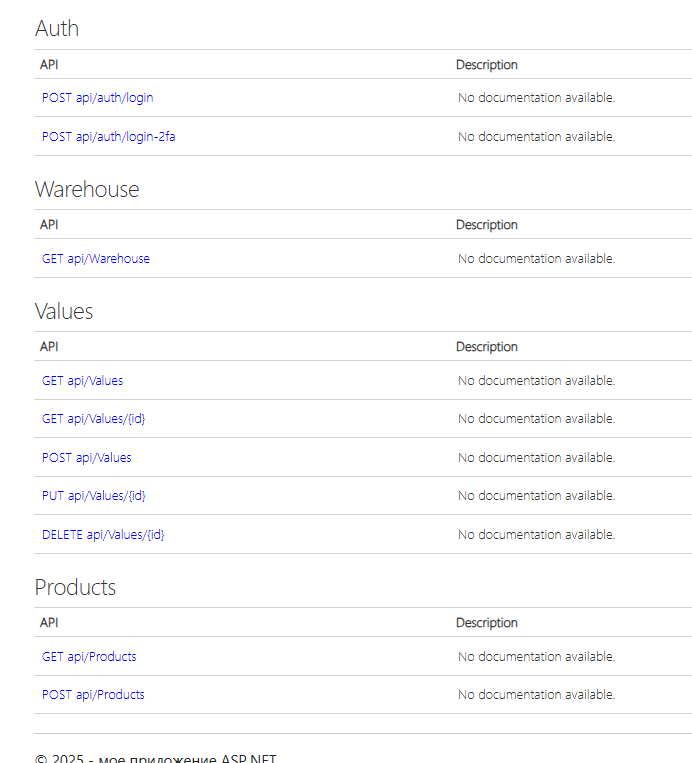


Рисунок 27 – Документация по API

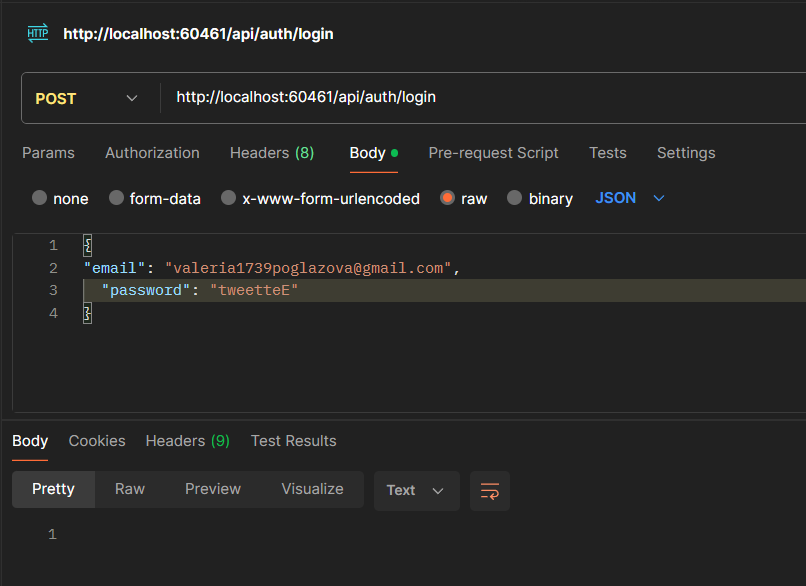


Рисунок 28 – Запрос в Postman (авторизация пользователя)

Тестирование интерфейса мобильного приложения.

*TestCase # 1*

Приоритет теста: Высокий

Название тестирования/Имя: Успешная двухфакторная аутентификация

Резюме испытания: Проверка успешной аутентификации пользователя с использованием двухфакторной аутентификации

Шаги тестирования: • Открыть приложение

• Ввести логин и пароль

• Ввести роль в системе

• Нажать "Войти"

Данные тестирования: Логин: valeria1739poglazova@gmail.com

Пароль: tweette

Ожидаемый результат: Пользователь успешно авторизован, открывается главное меню приложения

Фактический результат: Пользователь успешно авторизован, открывается главное меню приложения

Предпосылки: Пользователь зарегистрирован в системе

Постусловия: Пользователь находится в главном меню

Статус :(Pass/Fail) Pass

Комментарии: Тест пройден

*TestCase # 2*

Приоритет теста: Высокий

Название тестирования/Имя: Просмотр списка складов

Резюме испытания: Проверка отображения списка складов

Шаги тестирования: • Авторизоваться в приложении.

• Перейти в раздел "Склады".

• Проверить, правильно ли отображается информация

Данные тестирования: Список складов в базе данных

Ожидаемый результат: Отображается список складов с их названиями и адресами.

Фактический результат: Отображается список складов с их названиями и адресами.

Предпосылки: Пользователь авторизован в системе

Постусловия: Пользователь видит список складов

Статус :(Pass/Fail) Pass

Комментарии: Тест пройден

*Тест-кейс 3*

Приоритет теста: Высокий

Название тестирования/Имя: Форма ввода кода - Уведомление

Резюме испытания: Проверка корректного отображения уведомления

Данные тестирования: Неверный код

Ожидаемый результат: Отображается уведомление об ошибке.

Фактический результат: Отображается уведомление об ошибке.

Предпосылки: Пользователь авторизован в системе

Постусловия: Пользователь видит список товаров на складе

Статус (Pass/Fail): Pass

Комментарии: Тест пройден

*Тест-кейс 4*

Приоритет теста: Высокий

Название тестирования/Имя: Сканирование штрих кода

Резюме испытания: Проверка успешного сканирования штрих кода и отображения информации о товаре

Шаги тестирования: • Авторизоваться в приложении.

• Перейти в раздел "Сканирование".

• Сканировать штрих код товара.

Данные тестирования: Действительный штрих код товара

Ожидаемый результат: Отображается информация о товаре

Фактический результат : Отображается информация о товаре

Предпосылки: Пользователь авторизован в системе

Постусловия: Пользователь видит информацию о товаре

Статус (Pass/Fail): Fail

Комментарии: Тест не пройден

*TestCase # 5*

Приоритет теста: Высокий

Название тестирования/Имя :Ввод штрих кода вручную

Резюме испытания: Проверка штрих кода при вводе вручную и отображения информации о товаре

Шаги тестирования • Авторизоваться в приложении.

• Перейти в раздел "Сканирование".

• Нажать «Ввести штрих-код вручную»

• Ввести штрих код вручную и проверить, отображается ли информация о товаре

Данные тестирования: Действительный штрих код товара

Ожидаемый результат: Отображается информация о товаре

Фактический результат: Отображается информация о товаре

Предпосылки: Пользователь авторизован в системе

Постусловия: Пользователь видит информацию о товаре

Статус (Pass/Fail): Pass

Комментарии: Тест пройден

*Тест-кейс 6*

Приоритет теста: Высокий

Название тестирования/Имя: Просмотр профиля

Резюме испытания: Проверка отображения профиля пользователя с пропуском

Шаги тестирования: • Авторизоваться в приложении.

• Перейти в раздел "Профиль".

• Проверить, все ли отображается

Данные тестирования: Данные профиля пользователя

Ожидаемый результат: Отображается информация о пользователе и его пропуск.

Фактический результат: Отображается информация о пользователе и его пропуск.

Предпосылки: Пользователь авторизован в системе

Постусловия: Пользователь видит свой профиль

Статус (Pass/Fail): Pass

Комментарии: Тест пройден

Тестовое покрытие составило 10 тестов (20% отрицательных) на модуль.

Результаты тестирования подтвердили корректность работы программного модуля и его готовность к внедрению.

# 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:

* Модульное тестирование библиотеки:

Проведено тестирование методов библиотеки для работы с данными. Результаты показали, что 3 из 6 тестов были успешно пройдены. Тест, связанный с подсчетом общей стоимости товаров, завершился с ошибкой.

* Нагрузочное тестирование API:

Проведено тестирование API с использованием программы Postman. Результаты показали, что система стабильно справляется с нагрузкой в 100/1 000 запросов. Однако выявлено высокое время отклика для запросов GET, связанных с выводом списка товаров на складе (3116 мс).

* Тестирование интерфейса мобильного приложения:

Проведено тестирование пользовательского интерфейса мобильного приложения. Из 10 тестовых случаев 8 были успешно пройдены, а 2 завершились с ошибками (отсутствие переключателя видимости пароля и погрешность в отображении местоположения на карте).

1. Выявленные ошибки и рекомендации по их устранению:

* Ошибка в подсчете общей стоимости товаров:

Причина: Некорректная реализация метода GetTotalCost() в библиотеке.

Рекомендация: Пересмотреть логику расчета общей стоимости товаров и провести повторное тестирование.

* Высокое время отклика для запросов GET (список товаров на складе):

Причина: Недостаточная оптимизация запросов к базе данных.

Рекомендация: Оптимизировать SQL-запросы, добавить индексы и кэширование данных.

* Отсутствие переключателя видимости пароля:

Причина: Функционал не был реализован в интерфейсе авторизации.

Рекомендация: Добавить переключатель видимости пароля для улучшения удобства пользователей.

* Погрешность в отображении местоположения на карте:

Причина: Неточность в работе сервиса определения местоположения.

Рекомендация: Использовать более точные API для работы с геоданными (например, Google Maps API).

Оформленные отчеты по тестированию стали основой для дальнейшего улучшения системы. Результаты тестирования подтвердили, что система соответствует большинству требований технического задания, однако требуются доработки для устранения выявленных ошибок и оптимизации производительности.

# 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. Улучшение читаемости кода:

Код был переработан с учетом принципов чистого кода (Clean Code).

Добавлены комментарии к ключевым методам и классам для упрощения понимания логики.

Имена переменных, методов и классов были приведены в соответствие с общепринятыми стандартами именования (CamelCase, PascalCase).

Логика работы была разделена на более мелкие методы, что упростило понимание и тестирование кода.

1. Устранение дублирующихся участков кода:

Выявлены и устранены дублирующиеся фрагменты кода, такие как повторяющиеся SQL-запросы и методы обработки данных.

Для устранения дублирования были созданы универсальные методы, которые могут быть повторно использованы в разных частях системы.

1. Рефакторинг пользовательского интерфейса:

Улучшена структура XAML-файлов в настольном приложении за счет использования стилей и шаблонов.

1. Интеграция с системой контроля версий:

Изменения были зафиксированы в системе контроля версий Git и отправлены отдельными коммитами.

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.

# Разработка программного модуля для мобильного устройства

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

Разработан программный модуль для мобильного устройства, включающий:

1. Функция авторизации.

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

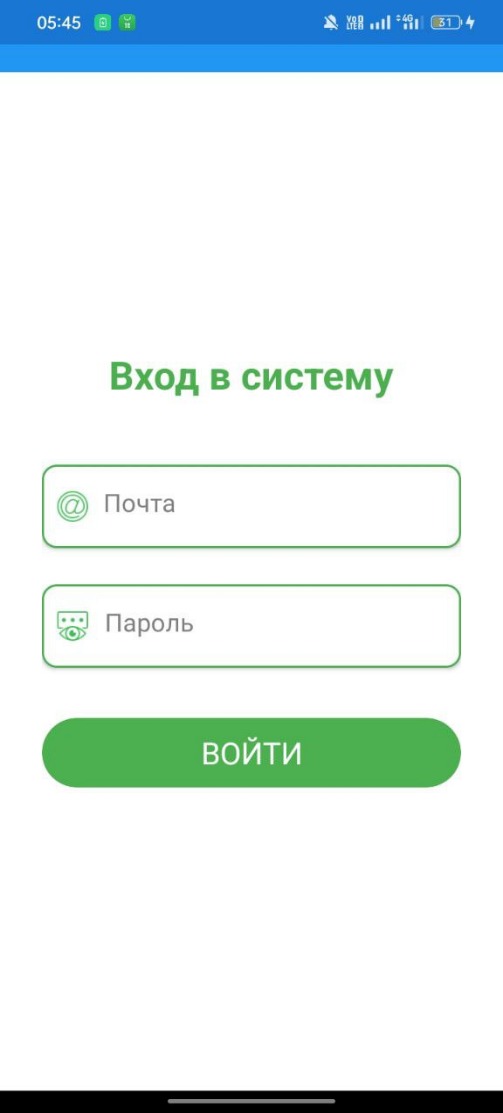


Рисунок 28 – Функция авторизации

1. Функция просмотра складов.

Отображает список доступных складов.

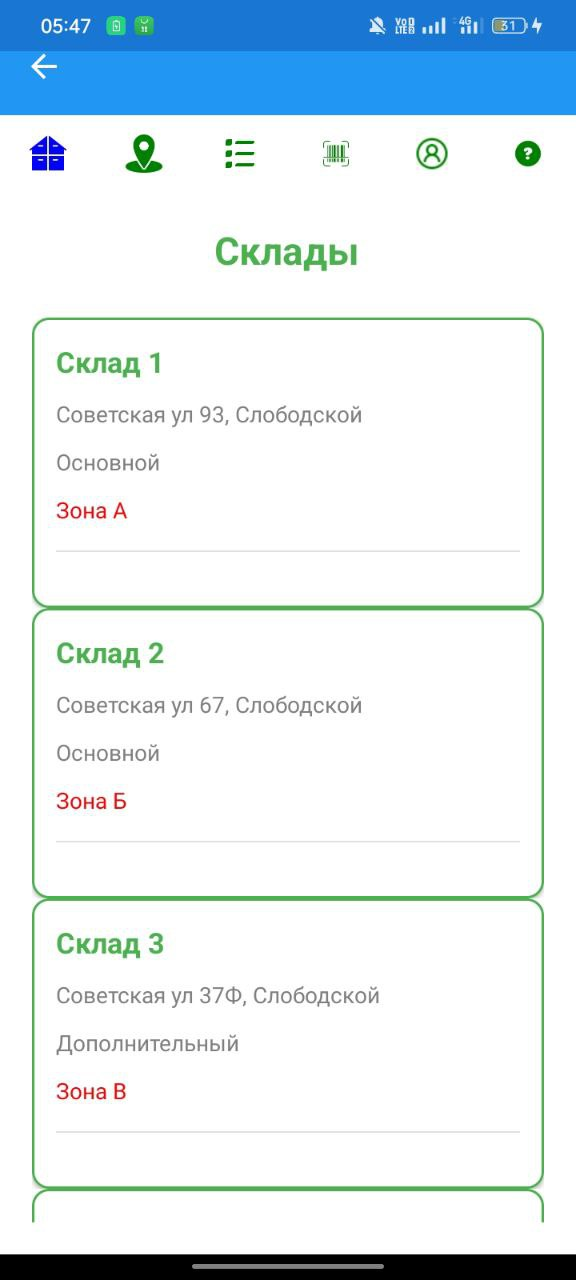


Рисунок 29 – Функция просмотра складов

1. Функция управления товарами.

Позволяет просматривать списки товаров на складе, выполнять поиск товаров, просматривать информацию о товарах.

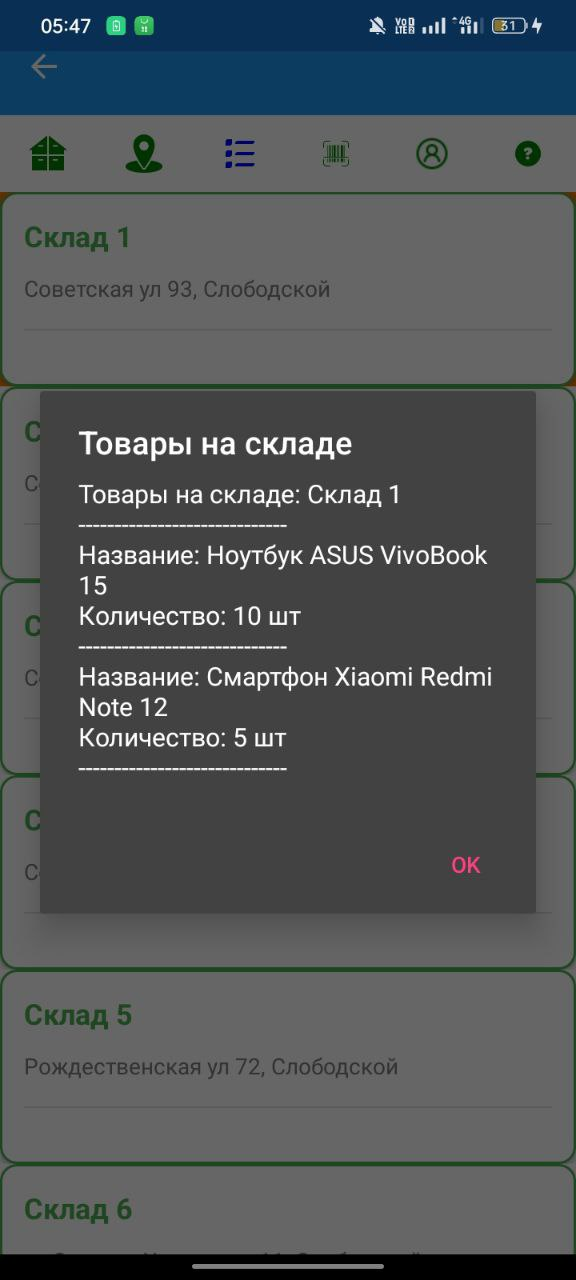


Рисунок 30 – Функция управления товарами

1. Функция использования карты.

Отображает местоположение складов на карте.

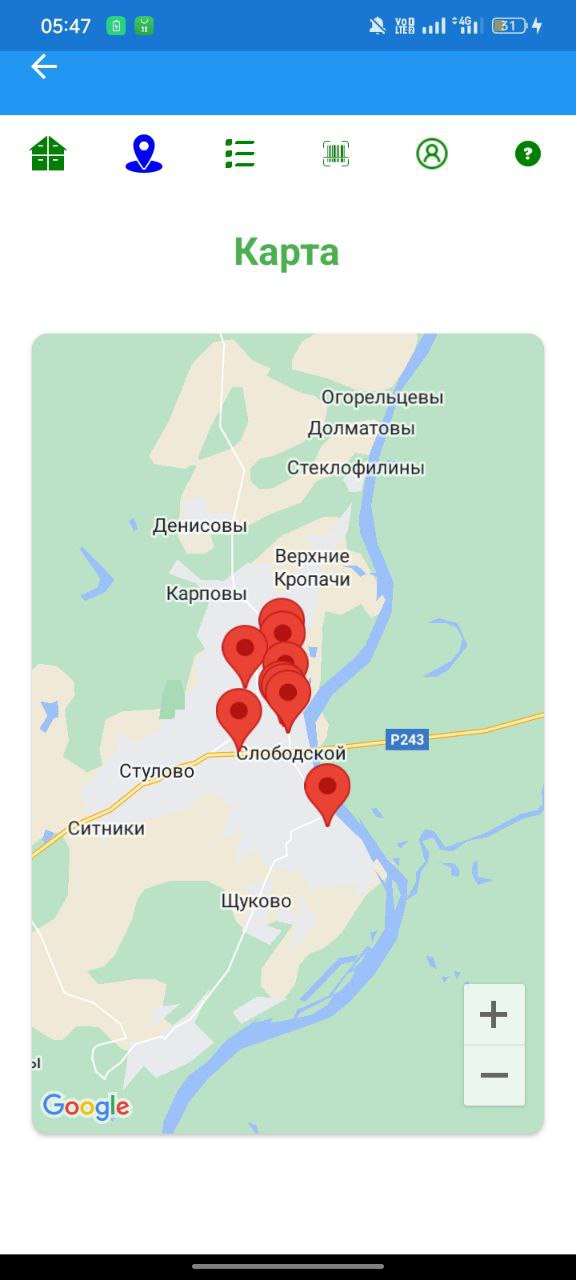


Рисунок 31 – Функция использования карты

1. Функция сканирования штрихкодов.

Позволяет выполнять сканирование штрихкодов для быстрого поиска товаров и выполнения складских операций.

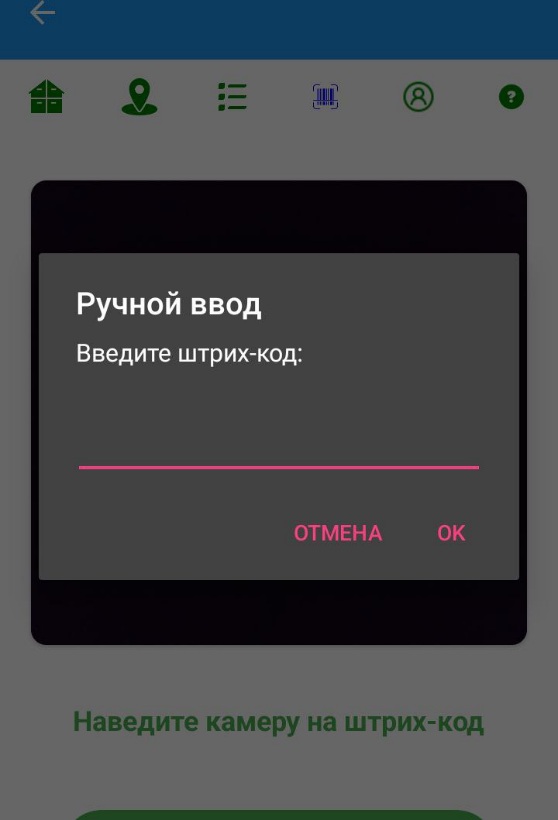
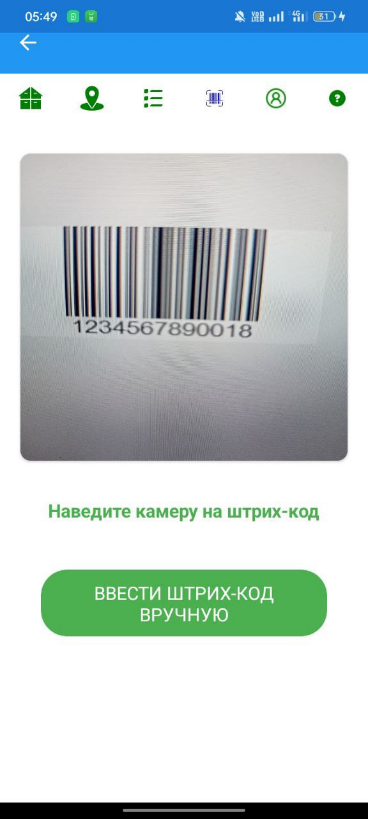


Рисунок 32 – Функция сканирования штрихкодов

# Проверка работоспособности программного модуля на мобильном устройстве или эмуляторе

Проверка работоспособности мобильного приложения проводилась на реальном устройстве Xiaomi Redmi Note с операционной системой Android 13. Устройство было выбрано как типичный представитель среднего ценового сегмента, что позволяет оценить производительность и стабильность приложения на большинстве современных смартфонов.

Протестированные функции приложения:

1. Двухфакторная аутентификация:

Проверена возможность входа в систему с использованием двухфакторной аутентификации (по коду).

1. Меню приложения:
2. Проверена корректность отображения всех пунктов меню:

Список складов и их местоположение.

Список товаров по складам.

Сканирование штрихкода (QR-кода).

Профиль пользователя.

Убедились, что переход между экранами происходит без задержек и ошибок.

1. Список складов и их местоположение:

Убедились, что местоположение складов отображается корректно на карте.

1. Список товаров по складам:

Проверена загрузка списка товаров для каждого склада.

Убедились, что данные о товарах (название, количество, категория) отображаются корректно.

Проверена работа фильтров и сортировки (по названию, количеству, категории).

1. Сканирование штрихкода (QR-кода):

Проверена работа сканера штрихкодов с использованием камеры устройства.

Убедились, что при сканировании товара отображается корректная информация о нем (название, количество, местоположение на складе).

Проверена обработка некорректных или поврежденных штрихкодов.

1. Профиль пользователя:

Убедились, что изменения сохраняются и синхронизируются с сервером.

1. Сохранение данных авторизации:

Проверена возможность сохранения данных авторизации для повторного входа без ввода логина и пароля.

1. Результаты тестирования:

Все функции приложения работают корректно на реальном устройстве.

Не выявлено критических ошибок или сбоев в работе.

Потребление памяти и энергии устройства находится в пределах нормы.

Мобильное приложение успешно прошло тестирование на реальном устройстве. Все функции работают стабильно, интерфейс интуитивно понятен, а производительность соответствует ожиданиям. Приложение готово к использованию в реальных условиях.

# Заключение

В ходе выполнения работы были успешно реализованы все этапы разработки программного модуля для системы складского учета. Начиная с анализа технического задания и проектирования архитектуры системы, заканчивая тестированием и оптимизацией, каждый этап был выполнен в соответствии с поставленными задачами и требованиями.

Разработанное приложение включает в себя:

1. API для взаимодействия с базой данных и обработки запросов.
2. Библиотеку для работы с данными, включая подсчет товаров, их стоимости и категорий.
3. Настольное приложение с двухфакторной аутентификацией, разграничением прав доступа и поддержкой локализации на два языка.
4. Мобильное приложение для удобной работы кладовщиков, включая сканирование штрихкодов и просмотр данных о складах и товарах.

Все компоненты системы прошли тщательное тестирование, включая модульное, интеграционное и нагрузочное тестирование. Результаты тестирования подтвердили стабильность и корректность работы приложения.

Ключевые достижения проекта:

1. Соответствие техническому заданию.

Все функциональные и нефункциональные требования были реализованы в полном объеме.

1. Удобство использования

Интерфейс приложения интуитивно понятен, что упрощает работу пользователей с разным уровнем подготовки.

1. Масштабируемость.

Архитектура системы позволяет легко добавлять новые функции и расширять функционал в будущем.

1. Кроссплатформенность.

Наличие настольного и мобильного приложения обеспечивает гибкость в использовании системы.

Проект завершен успешно, и система готова к использованию в реальных условиях. Разработанное приложение не только соответствует всем требованиям технического задания, но и предоставляет удобный и надежный инструмент для автоматизации складского учета. Благодаря модульной архитектуре и поддержке современных технологий, система имеет большой потенциал для дальнейшего развития и масштабирования.

# Приложения к отчету

Ссылка на GitHub, где расположены все материалы по учебной практике ПМ.01 -https://github.com/Valeria12-10/UchPracricPM01.