|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное г**о**сударственное АВТОНОМНОЕ образовательное учреждение высшего образования  «Национальный исследовательский ядерный университет «МИФИ» |
| **Димитровградский инженерно-технологический институт –**  филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»  **(ДИТИ НИЯУ МИФИ)** |

**Специальность 09.02.07 Информационные системы и программирование**

**ОТЧЕТ**

**по производственной практике**

**(по профилю специальности)**

|  |  |
| --- | --- |
| **Наименование практики** | ПП.01.01 Производственная практика |
| **Профессиональный модуль:** | ПМ.01 Разработка модулей программного обеспечения для компьютерных систем |
| **Обучающийся** | Степанова Дарья Дмитриевна |
| **Группа** | 331 |
| **Место проведения практики:** | ООО «АИС Город» |
| **Сроки прохождения  практики:** | с 12.05.2025 г. по 17.05.2025 г.,  с 26.05.2025 г. по 07.06.2025 г. |
| **Руководитель практики от**  **организации/предприятия**  **базы практики:** | Наскальнюк Андрей Николаевич |
| **Руководитель практики от**  **образовательной**  **организации:** | Романов Василий Анатольевич |
| **Итоговая оценка** |  |
| **Дата** | 07.06.2025 |

Димитровград, 2025

**Содержание**

[ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОХОЖДЕНИЕ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ 3](#_Toc197773842)

[1 Задание 1 Проведение инструктажа по технике безопасности. Ознакомление с предприятием. Установка инструментальных программных средств разработки программного обеспечения. 3](#_Toc197773843)

[1.1 Проведение инструктажа по технике безопасности. 3](#_Toc197773844)

[1.3 Установка и настройка среды программирования 4](#_Toc197773845)

[1.4 Установка инструментария и настройка среды для разработки мобильных приложений 7](#_Toc197773846)

[1.5 Установка и настройка системы контроля версий 10](#_Toc197773847)

[2 Задание 2 Разработка прикладной программы 11](#_Toc197773848)

[2.1 Разработка прикладной программы для работы со связанными таблицами с использованием Visual C# и технологии ADO.NET 11](#_Toc197773849)

[2.2 Создание приложения для работы с БД, создание запросов к БД 21](#_Toc197773850)

[3 Задание 3 Тестирование программного обеспечения 23](#_Toc197773851)

[3.1 Рефакторинг программного кода приложения 23](#_Toc197773852)

[3.2 Разработка тестов. Отладка и тестирование программы на уровне модуля 24](#_Toc197773853)

[3.3 Анализ результатов тестирования. Тестирование с помощью инструментов среды разработки 24](#_Toc197773854)

[4 Задание 4 Разработка мобильного приложения 24](#_Toc197773855)

[4.1 Создание интерфейса мобильного приложения 25](#_Toc197773856)

[4.2 Тестирование и оптимизация мобильного приложения 25](#_Toc197773857)

[5 Задание 5 Задание работодателя 25](#_Toc197773858)

[6 Задание 6 Оформление документации. Разработка презентации 25](#_Toc197773859)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc197773860)

[Заключение о прохождении производственной практики 28](#_Toc197773861)

# **ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОХОЖДЕНИЕ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

**1 Задание 1 Проведение инструктажа по технике безопасности. Ознакомление с предприятием. Установка инструментальных программных средств разработки программного обеспечения.**

**1.1 Проведение инструктажа по технике безопасности.**

В соответствии с установленными требованиями перед началом прохождения практики был проведён инструктаж по технике безопасности. Инструктаж прошёл 12.05.2025. Провёл инструктаж Наскальнюк Андрей Николаевич.

В ходе инструктажа были рассмотрены основные требования по обеспечению безопасных условий труда на рабочем месте. Особое внимание было уделено следующим вопросам:

* Эргономика рабочего места;
* Исправность оборудования;
* Действиям при возникновении чрезвычайных ситуаций;
* Организация рабочего места.

**1.2** **Программное обеспечение и автоматизированных систем предприятия (организации).**

Предприятие «АИС Город» специализируется на комплексной автоматизации и цифровизации процессов городского управления и взаимодействия с жителями. Основными направлениями деятельности системы являются: централизованный учет и расчеты в сфере жилищно-коммунального хозяйства, организация электронных сервисов для населения (личные кабинеты, прием обращений), управление городской инфраструктурой и ресурсами, формирование аналитической отчетности для муниципальных органов, поддержка электронного документооборота, а также интеграция данных различных городских служб.

Автоматизация бизнес-процессов

Основные направления автоматизации бизнес-процессов включают:

* Автоматизация расчетов и начислений за жилищно-коммунальные услуги (ЖКХ);
* Электронное взаимодействие с жителями (Личный кабинет, обращения);
* Управление ресурсами и инфраструктурой города;
* Электронный документооборот (ЭДО);
* Формирование аналитической отчетности и поддержки принятия решений.

Требования к программному обеспечению и оборудованию рабочего места

* Средства вычислительной техники, на которых осуществляется штатное функционирование СКЗИ, должны быть оборудованы средствами контроля за их вскрытием (опечатаны, опломбированы).
* Место опечатывания (опломбирования) СКЗИ, аппаратных средств должно быть таким, чтобы его можно было визуально контролировать.

Методы организации командной работы

Работа в «АИС Город» организована по принципу **межведомственного взаимодействия и формирования кросс-функциональных рабочих групп, каждая из которых включает специалистов по работе с населением, системных администраторов, аналитиков данных, координаторов взаимодействия с поставщиками услуг, а также представителей профильных городских служб (например, ЖКХ, благоустройство).**

Основной методологией является адаптивный подход, ориентированный на оперативное решение задач и постоянное улучшение. Он включает регулярные координационные совещания, планирование краткосрочных циклов задач (например, по обработке сезонных обращений или внедрению новых функций) и демонстрации достигнутых результатов для всех заинтересованных сторон. Также активно используется Kanban для управления потоком обращений граждан, оперативной поддержки пользователей системы и систематизации текущих задач по обслуживанию и развитию АИС.

**1.3 Установка и настройка среды программирования**

**Постановка задачи:**

Установить Visual Studio на свой ПК.

**Выполнение:**

Официальный установщик Visual Studio загрузила с сайта https://visualstudio.microsoft.com/ru/vs/

Запуск установщика и выбор версии

После загрузки установщика выбрала версию Visual Studio Community 2022, как бесплатная и полнофункциональная среда для студентов и индивидуальных разработчиков.

На этапе установки выбрала рабочий набор компонентов:

* разработка классических приложений .NET;
* хранение и обработка данных (для поддержки решений по обработке данных с помощью SQL Server).

После выбора компонентов началась автоматическая загрузка и установка необходимых файлов.

После завершения установки выполнила первый запуск Visual Studio.

Для проверки корректности установки создала тестовый проект на C# (Windows Forms .NET Framework). Компиляция и запуск прошли успешно.

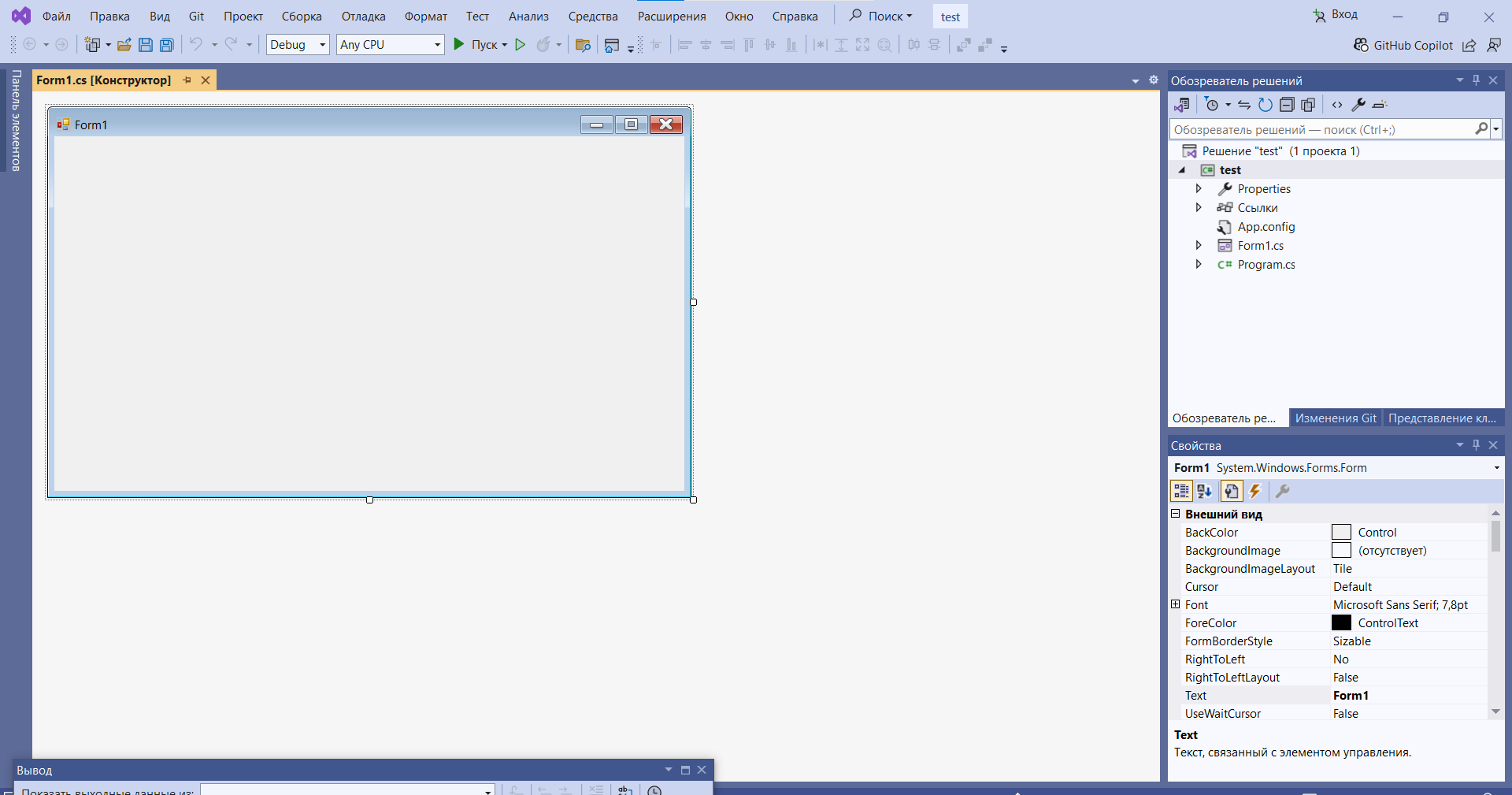


Рисунок 1.2 – Тестовый проект

**1.4 Установка инструментария и настройка среды для разработки мобильных приложений**

**Задание**

**Постановка задачи:**

Установить Android Studio на свой ПК.

**Выполнение:**

Установочный файл Android Studio загрузила с официального сайта https://developer.android.com/studio.

При запуске установщика выбрала следующие компоненты:

* android studio IDE;
* android virtual device.

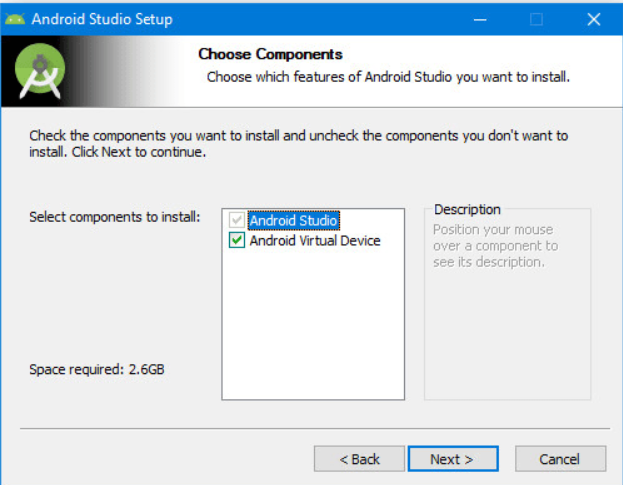


Рисунок 1.3 - Выбор компонентов

На следующем шаге выбрала тип установки – Standard, рекомендованный для большинства пользователей. Он включает оптимальные настройки и компоненты по умолчанию.

Через мастер создания проекта выбрала шаблон проекта – Empty Activity. Этот шаблон содержит пустую активность, с которой удобно начинать разработку.

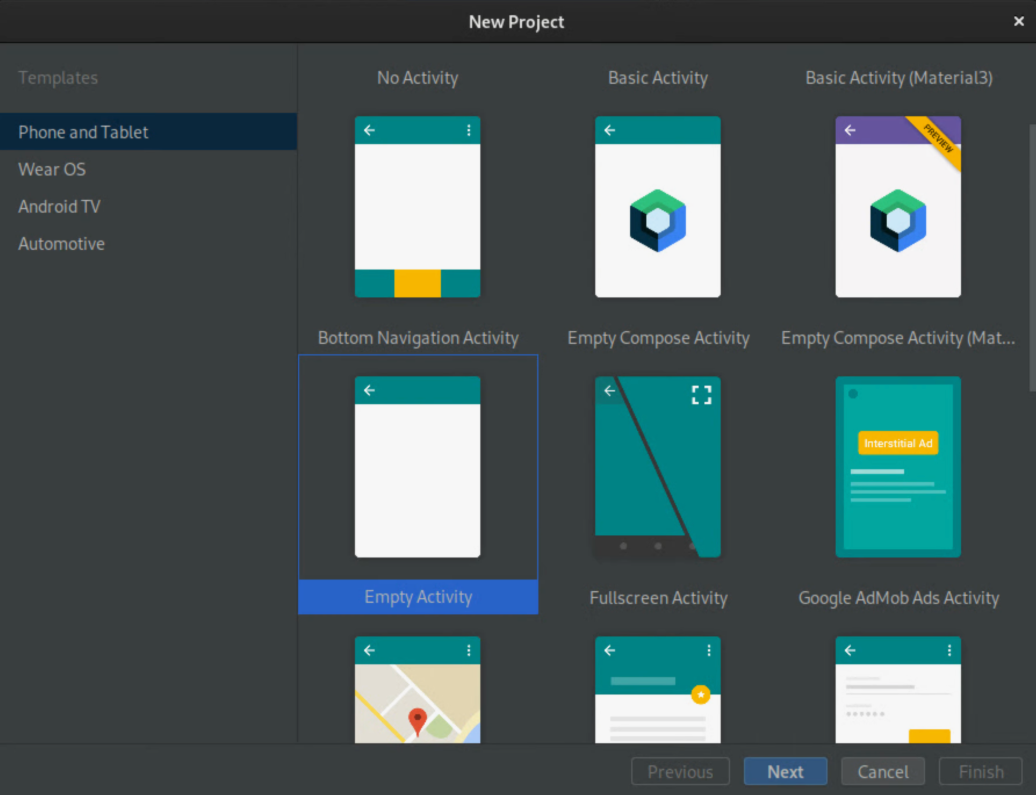


Рисунок 1.4 – Выбор шаблона Empty Activity

На следующем экране задала параметры проекта:

* Имя проекта: testApplication;
* Язык: Java;
* Минимальная версия SDK: API 24 (Android 7.0);

После создания проекта запустила его в Android Studio.

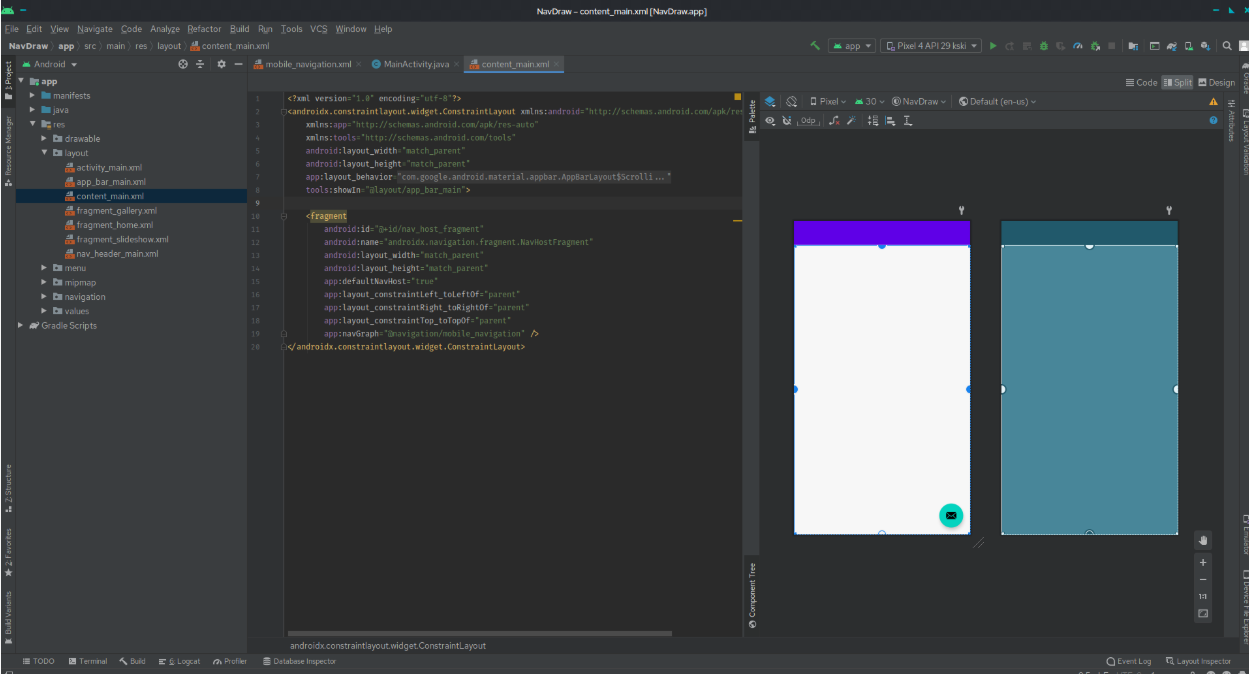


Рисунок 1.5 – Окно Android Studio

**1.5 Установка и настройка системы контроля версий**

Система контроля версий (СКВ) представляет собой инструмент для отслеживания изменений в программном коде и других файлах проекта. Она позволяет управлять историей изменений, работать в команде, синхронизировать изменения между несколькими разработчиками, а также возвращаться к предыдущим версиям кода при необходимости.

Основные задачи системы контроля версий:

* хранение всех изменений в проекте;
* совместная работа над кодом;
* создание веток и управление слияниями;
* обеспечение резервного копирования и отката изменений.

Типы систем контроля версий:

* централизованные (CVS, Subversion) – один сервер, все действия идут через него;
* распределённые (Git, Mercurial) – каждый разработчик имеет локальную копию репозитория.

Git – наиболее популярная распределённая система контроля версий, созданная Линусом Торвальдсом. Git обеспечивает высокую скорость, надёжность, параллельную работу и гибкость в управлении ветками.

**Задание**

**Постановка задачи:**

Установить Git Bush на свой ПК.

**Выполнение:**

Установочный файл загрузила с официального сайта <https://git-scm.com/downloads>. После запуска установщика выбрала параметры по умолчанию.

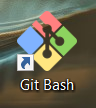


Рисунок 1.6 - Установленное приложение Git

После установки выполнила базовые настройки Git с помощью команд, показанных на рисунке 1.7. Эти данные используются при коммитах для указания автора.

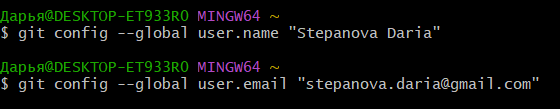


Рисунок 1.7 - Настройка имени пользователя и почты в Git

Для инициализации репозитория в рабочей папке использовала команду, представленную на рисунке 1.8. Эта команда создает скрытую папку .git, в которой хранится локальный репозиторий.

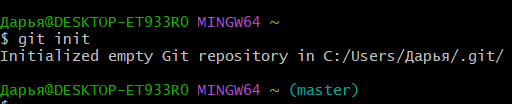


Рисунок 1.8 - Создание локального репозитория Git

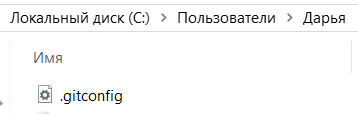


Рисунок 1.9 – Созданный локальный репозиторий

**2 Задание 2 Разработка прикладной программы**

**2.1 Разработка прикладной программы для работы со связанными таблицами с использованием Visual C# и технологии ADO.NET**

Описание предметной области

Предметная область учета заявок на ремонт оборудования охватывает все процессы, связанные с фиксацией, обработкой, выполнением и контролем устранения неисправностей в различных видах оборудования, используемого на предприятии. Она является критически важной для обеспечения бесперебойной работы, минимизации простоев и эффективного управления активами.

Основные цели предметной области:

* Обеспечение оперативного реагирования: Быстрая регистрация и передача информации о неисправности соответствующим службам.
* Оптимизация процессов ремонта: Упорядочивание всех этапов от подачи заявки до завершения ремонта.
* Повышение эффективности использования оборудования: Сокращение времени простоя и продление срока службы активов.
* Управление ресурсами: Эффективное планирование загрузки ремонтного персонала, учет расхода запасных частей и материалов.
* Сбор и анализ данных: Накопление информации о неисправностях, их причинах, стоимости и сроках устранения для дальнейшего анализа и принятия управленческих решений (например, для предиктивного обслуживания или закупки нового оборудования).

**Техническое задание**

1. Общие сведения
   1. Наименование проекта: Разработка программного модуля для учета заявок на ремонт оборудования.
   2. Заказчик: ООО «Техносервис».
   3. Исполнитель: Компания «IT-Решения-Степанова Дарья Дмитриевна».
2. Функциональные требования
   1. Возможность добавления заявок в базу данных с указанием следующих параметров:

* Номер заявки;
* Дата добавления;
* Оборудование, которое требует ремонта;
* Тип неисправности;
* Описание проблемы;
* Клиент, который подал заявку;
* Статус заявки (в ожидании, в работе, выполнено).
  1. Возможность редактирования заявок:
* Изменение этапа выполнения (выполнено, в работе, не выполнено);
* Изменение описания проблемы;
* Изменение, ответственного за выполнение работ.
  1. Возможность отслеживания статуса заявки:
* Отображение списка заявок;
* Получение уведомлений о смене статуса заявки;
* Поиск заявки по номеру или по параметрам.
  1. Возможность отслеживания статуса заявки клиентом, который подал заявку, с помощью мобильного приложения:
* Отображение списка заявок;
* Получение уведомлений о смене статуса заявки;

Поиск заявки по номеру или по параметрам.

* 1. Возможность назначения ответственных за выполнение работ:
* Добавление исполнителя к заявке;
* Отслеживание состояния работы и получение уведомлений о ее завершении;
* Исполнитель может добавлять комментарии на форме заявки.
  1. Расчет статистики работы отдела обслуживания:
* Количество выполненных заявок;
* Среднее время выполнения заявки;
* Статистика по типам неисправностей.

1. Нефункциональные требования
   1. Кроссплатформенность:

* Поддержка работы на ОС семейства Windows.
  1. Безопасность:
* Логин и пароль для доступа к приложению;
* Доступ к данным должен быть ограничен в зависимости от роли пользователя.
  1. Удобство использования:
* Простой и интуитивный интерфейс;
* Информативные уведомления и подсказки.
  1. Производительность:
* Приложение должно иметь быстрый доступ к данным;
* Минимальное время отклика на запросы пользователя.

1. Требования к реализации
   1. Языки программирования: Visual Studio C#, Android Studio;
   2. СУБД: MS SQL.

**Постановка задачи:**

Создать приложение сопровождать скринами и описанием. Разместить разработанную программу на Github.com с указанным адресом удаленного репозитория.

**Выполнение:**

В соответствии с анализом предметной области и требованиями технического задания, разработала прикладную программу для учёта заявок на ремонт оборудования.

Создала проект типа Windows Forms .NET Framework в Visual Studio на языке C#. Для взаимодействия с базой данных использовала технологию ADO.NET.

Реализовала форму авторизации пользователя (loginForm.cs) со следующими компонентами:

* label – заголовок и подписи к полям;
* textBox – для ввода логина и пароля;
* checkBox – «Запомнить меня»;
* button – «Войти»

Реализовала функционал на форме авторизации:

* проверка введённых данных;
* получение из БД роли пользователя;
* запоминание логина/пароля.

Результат проделанной работы представила на рисунке 2.1.1.

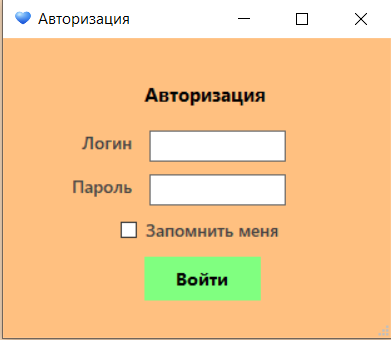


Рисунок 2.1 – Форма авторизации

Реализовала главную форму приложения (mainForm.cs) со следующими компонентами:

* tabControl – для навигации между разделами;
* dataGridView – отображение данных каждой сущности;
* comboBox, textBox – фильтрация и поиск записей;
* button – добавление, изменение, удаление записей.

Реализовала функционал на главной форме:

* загрузка и отображение данных из связанных таблиц;
* поиск и фильтрация записей;
* разграничение доступа к действиям добавления, редактирования, удаления в зависимости от роли пользователя.

Результат проделанной работы представила на рисунке 2.2

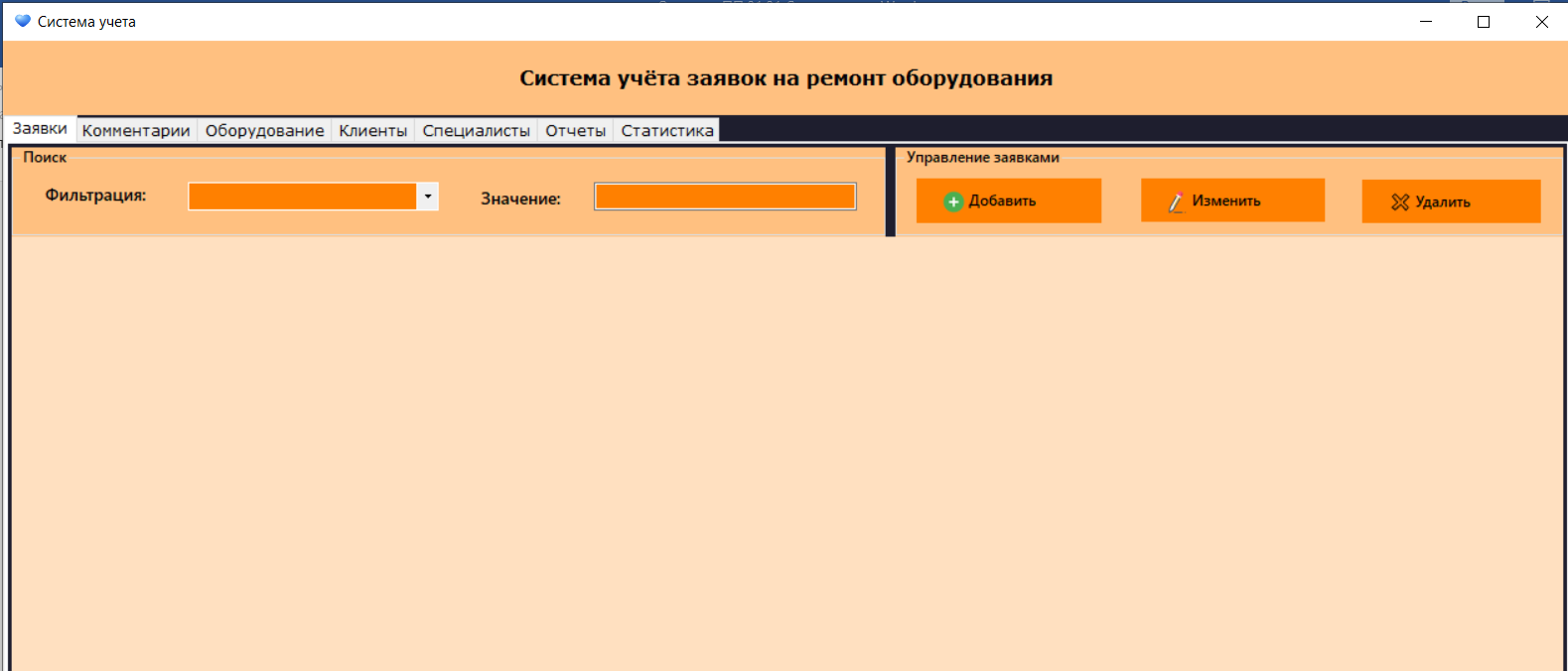


Рисунок 2.2 – Главная форма

Реализовала форму добавления заявки (addRequest.cs) со следующими компонентами:

* comboBox – выбор клиента, оборудования, типа ремонта, статуса, специалиста;
* textBox – поле описание, выбор даты заявки;
* button – добавить запись.

Реализовала функционал на форме добавления заявки:

* все поля обязательны для заполнения;
* в выпадающих списках отображаются значения из связанных таблиц;
* после нажатия кнопки «Добавить» данные сохраняются в базу данных.

Результат проделанной работы представила на рисунке 2.3.

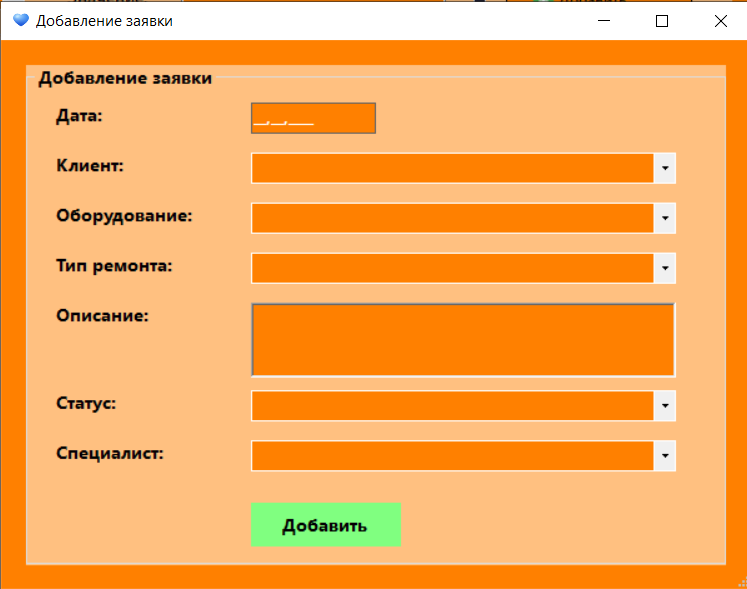


Рисунок 2.3 - Форма добавления заявки

Реализована форма редактирования заявки. Компоненты аналогичны форме добавления. Реализован функционал:

* поля предварительно заполняются текущими значениями из выбранной заявки;
* возможность изменить статус заявки, описание, специалиста и т.д.;
* сохранение изменений в базу.

Реализовала форму управления клиентами со следующими компонентами:

* dataGridView – отображение списка клиентов;
* textBox – для ввода и редактирования данных клиента (ФИО, телефон, e-mail и др.);
* button – добавление, изменение, удаление записи.

Реализовала функционал на формах управления клиентами:

* возможность добавления нового клиента в базу;
* возможность редактирования информации о клиенте;
* удаление клиента при необходимости.

Результат проделанной работы представила на рисунке 2.4.

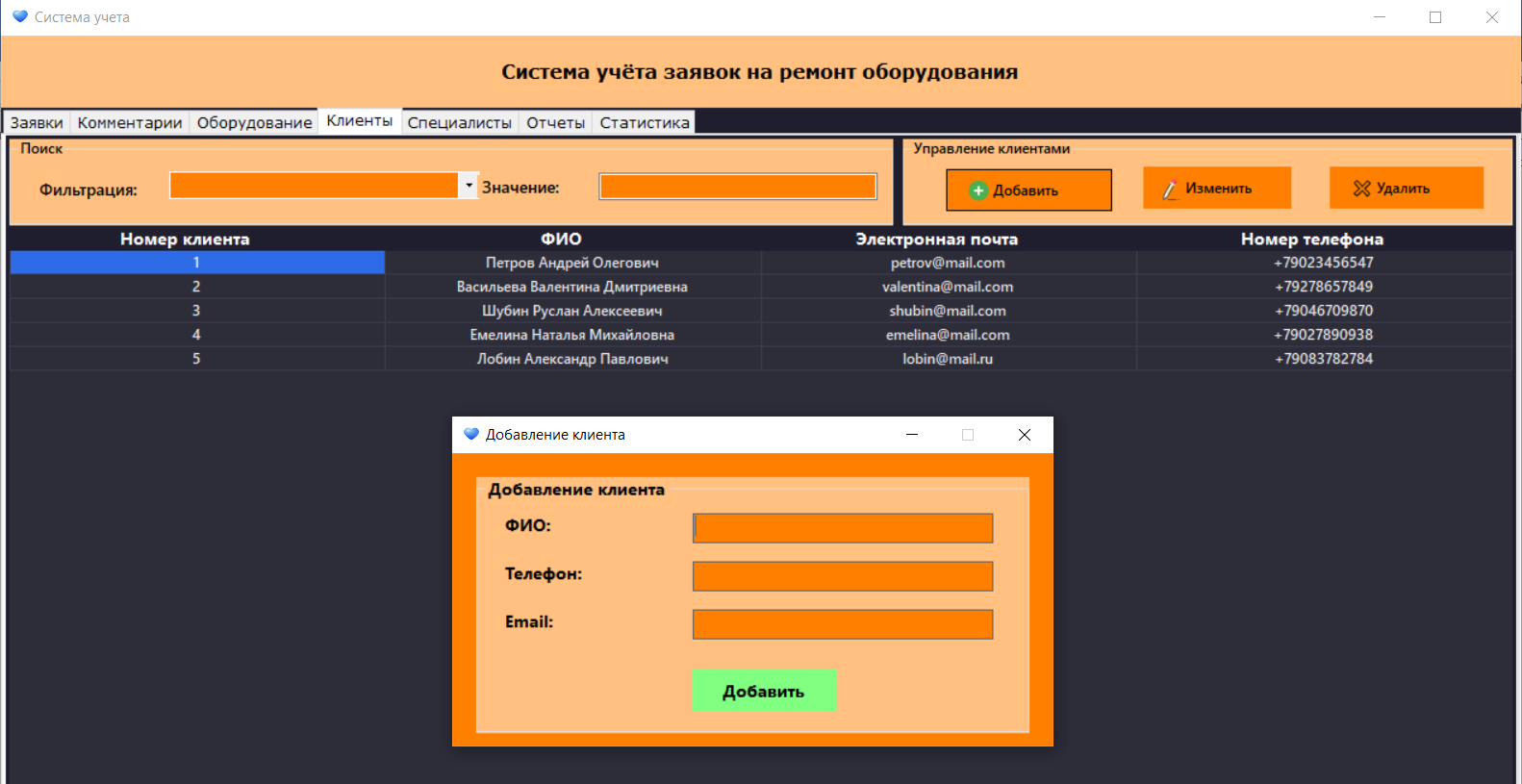


Рисунок 2.4 – Управление клиентами

Реализовала форму управления оборудованием со следующими компонентами:

* dataGridView – отображение списка оборудования;
* textBox – для ввода и редактирования характеристик оборудования (наименование, модель, серийный номер и др.);
* button – добавление, изменение, удаление записи.

Реализовала функционал на формах управления оборудованием:

* возможность добавления нового оборудования в базу;
* редактирование информации об оборудовании;
* удаление оборудования.

Результат проделанной работы представила на рисунке 2.5.

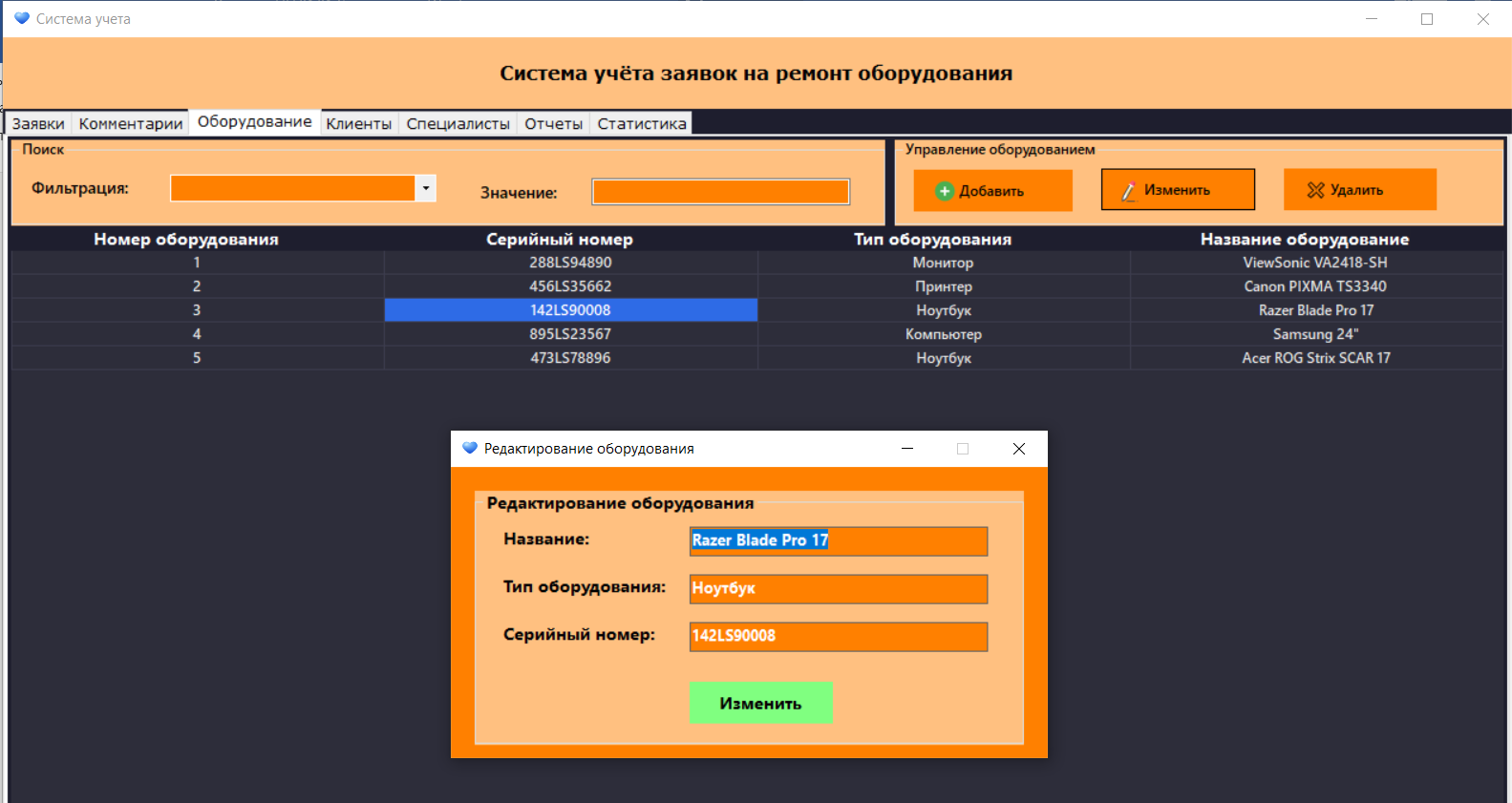


Рисунок 2.5 – Управление оборудованием

Реализовала форму управления специалистами со следующими компонентами:

* dataGridView – отображение списка специалистов;
* textBox – для ввода и редактирования данных специалиста (ФИО, должность, контактные данные);
* button – добавление, изменение, удаление записи.

Реализовала функционал на формах управления специалистами:

* возможность добавления нового специалиста;
* редактирование сведений о специалистах;
* удаление специалиста.

Результат проделанной работы представила на рисунке 2.6.

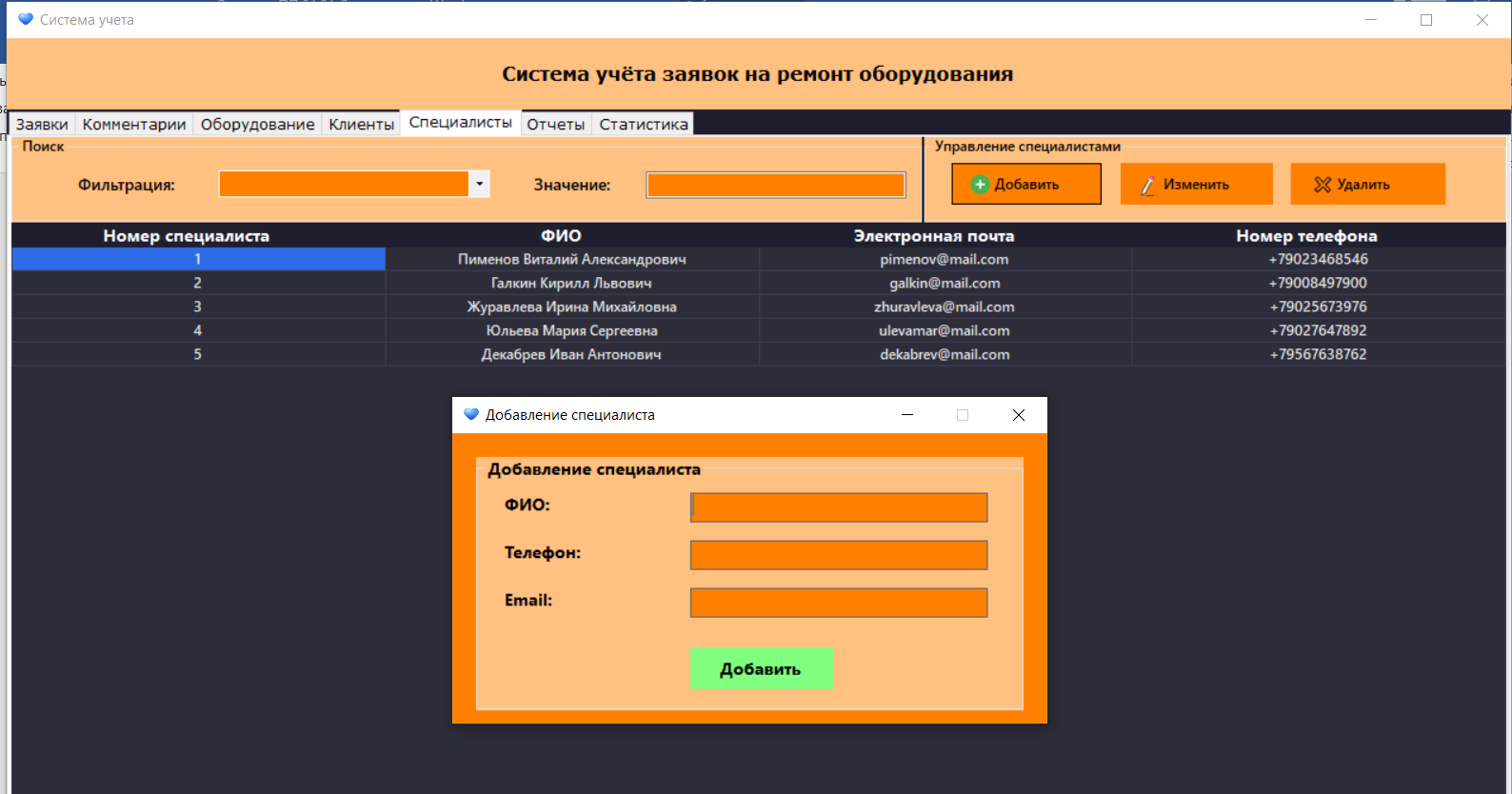


Рисунок 2.6 – Управление справочником специалистов

Реализовала форму добавления и редактирования комментариев со следующими компонентами:

* dataGridView – отображение списка комментариев;
* comboBox – выбор связанной заявки и специалиста;
* textBox – ввод текста комментария, выбор даты комментария;
* button – добавить, изменить, удалить комментарий.

Реализовала функционал на формах комментариев:

* возможность добавления нового комментария к заявке;
* возможность редактирования и удаления комментария;
* отображение всех комментариев с привязкой к заявкам и специалистам.

Результат проделанной работы представила на рисунке 2.7.

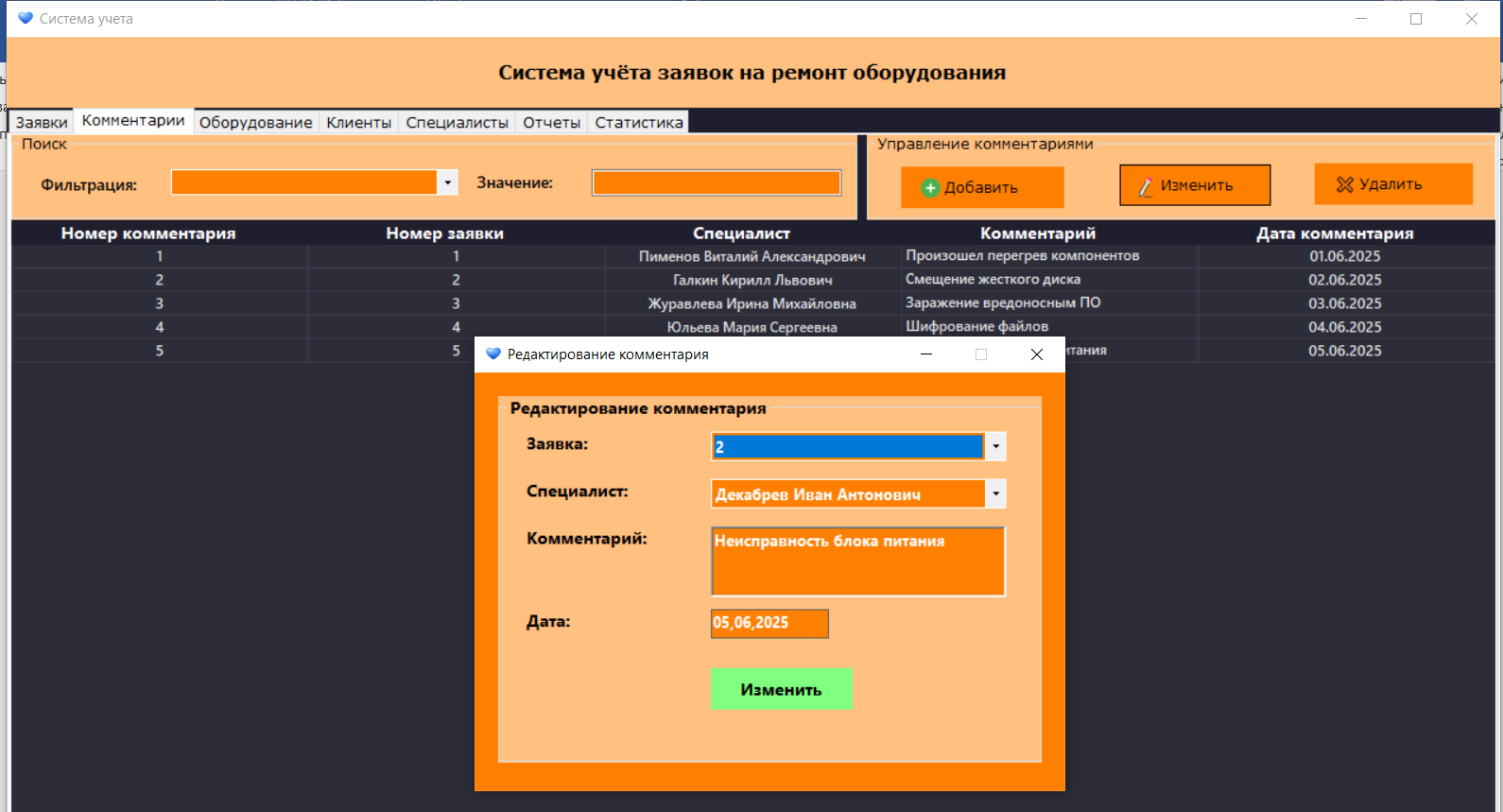


Рисунок 2.7 – Формы с комментариями к заявкам

Реализовала форму формирования отчётов и просмотра статистики со следующими компонентами:

* chart – построение круговой диаграммы по типам неисправностей;
* dataGridView – отображение сводных данных (количество заявок, среднее время выполнения и др.);
* label – вывод ключевых показателей (количество выполненных заявок, среднее время выполнения).

Реализовала функционал для аналитики и отчётности:

* автоматическое формирование статистики на основе данных из базы;
* визуализация данных с помощью диаграмм;
* быстрый просмотр и анализ показателей эффективности работы.

Разместила разработанную программу на платформе GitHub. Для этого был создан удалённый репозиторий, в который загружены все исходные коды проекта, включая формы, классы, конфигурационные файлы и необходимые ресурсы.

Адрес удалённого репозитория:

https://github.com/Daria-Stepanova-blip/PP.01.01Stepanova331

Результат проделанной работы представлен на рисунке 2.8.

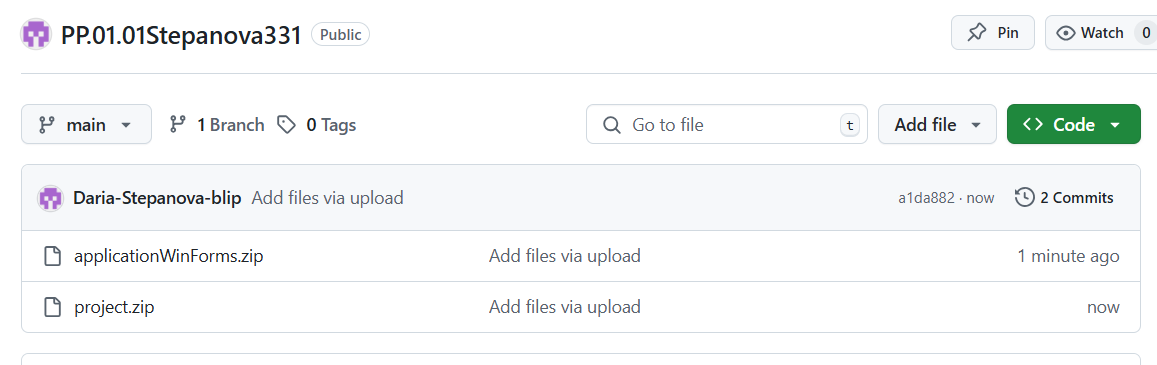


Рисунок 2.8 – Размещение исходного кода программы на github

**2.2 Создание приложения для работы с БД, создание запросов к БД**

Создание базы данных и прикладного программного обеспечения для работы с ней – важный этап при разработке информационных систем. Процесс включает:

* Сбор и анализ требований: Определение, какие данные будут храниться, для каких целей, кто будет их использовать и какие операции будут выполняться.
* Проектирование структуры (моделирование): Разработка логической модели данных (сущности, их атрибуты, связи между ними). Определяются таблицы, поля, типы данных, первичные и внешние ключи. Часто используется нормализация для исключения избыточности.
* Выбор системы управления базами данных (СУБД): Выбор конкретной СУБД (например, MySQL, PostgreSQL, MS SQL Server, Oracle, MongoDB) в зависимости от требований к производительности, масштабируемости, функционалу и бюджету.
* Реализация (создание схемы): Физическое создание базы данных и всех ее объектов (таблиц, индексов, представлений, хранимых процедур) в выбранной СУБД с использованием языка определения данных (DDL).
* Заполнение данными: Внесение начальных или тестовых данных в созданные таблицы.
* Тестирование и оптимизация: Проверка корректности работы, производительности запросов, безопасности данных. При необходимости – оптимизация структуры или запросов.

В рамках предметной области «Учёт заявок на ремонт оборудования» выделила 9 сущностей, включая:

* client – клиенты, подающие заявки;
* equipment – обслуживаемое оборудование;
* fault\_type – типы неисправностей;
* request – заявки на ремонт;
* status – статусы заявок;
* specialist – ответственные за ремонт;
* report – отчёты о проделанных работах;
* comment – комментарии исполнителей;
* user – учётные записи с ролями.

Между таблицами установила внешние ключи, обеспечивающие целостность данных и логические связи. Например, каждая заявка request связана с клиентом (client\_id), специалистом (specialist\_id), оборудованием (equipment\_id) и имеет статус (status\_id).

На основании структуры создала таблицы с типами данных в SQL Server Management Studio. Использовались следующие типы данных: int, nvarchar, float, date.

Для взаимодействия приложения с базой данных в составе проекта реализовала отдельный класс dateBase.cs.

Также создала вспомогательный класс data, хранящий актуальные значения текущей роли пользователя, идентификаторов выбранных записей (заявки, оборудования, клиента, специалиста и других сущностей), что обеспечивает передачу данных между формами и реализацию ролевого доступа.

Пример работы с таблицами и визуализации данных в интерфейсе привела на рисунках 2.9 и 2.10.

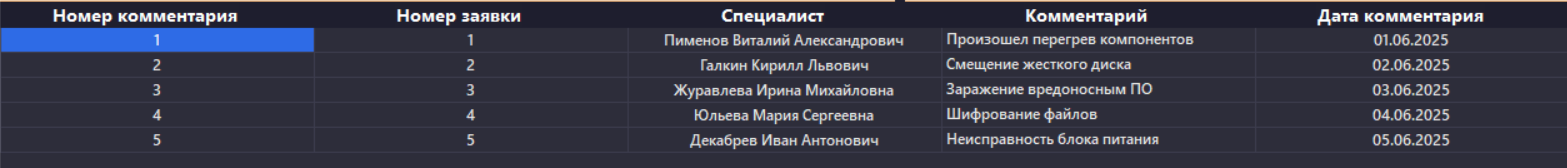


Рисунок 2.9 – Отображение данных таблицы «Комментарии»

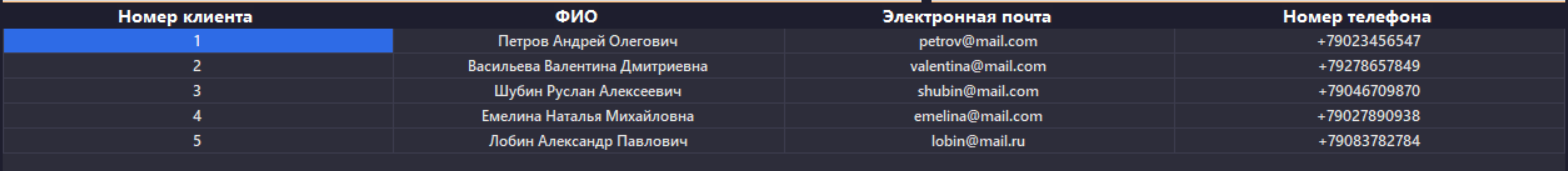


Рисунок 2.10 – Отображение данных таблицы «Клиенты»

**3 Задание 3 Тестирование программного обеспечения**

**3.1 Рефакторинг программного кода приложения**

Рефакторинг – это систематическая переработка кода, направленная на повышение его качества, читаемости и простоты сопровождения без добавления новой функциональности или изменения логики работы приложения.

Основные приёмы рефакторинга

* Извлечение метода (Extract Method): Выделение блока кода в новый, отдельно названный метод. Улучшает читаемость и устраняет дублирование.
* Встраивание метода (Inline Method): Перемещение тела метода в место его вызова и удаление самого метода. Применяется, когда метод слишком прост или его имя не добавляет ясности.
* Переименование (Rename): Присвоение более понятного и выразительного имени переменной, методу, классу или любому другому элементу кода.
* Введение поясняющей переменной (Introduce Explaining Variable): Создание временной переменной для хранения результата сложного выражения, делая его более читабельным.
* Перемещение метода/поля (Move Method/Field): Перенос метода или поля в класс, где оно логически более уместно и имеет больше связей с данными этого класса.
* Извлечение класса (Extract Class): Выделение части функциональности или данных из большого класса в новый, отдельный класс. Помогает уменьшить размер класса и повысить его связность.
* Декомпозиция условного выражения (Decompose Conditional): Разделение сложного условного оператора (if/else или switch) на отдельные методы для каждого условия или действия.
* Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism): Использование наследования и полиморфизма для устранения сложных if/else или switch конструкций, которые зависят от типа объекта.
* Удаление дублирующегося кода (Remove Duplicate Code): Использование методов извлечения, обобщения или шаблонов проектирования для устранения повторяющихся блоков кода в разных местах.
* Удаление "мёртвого" кода (Remove Dead Code): Устранение неиспользуемых методов, полей, классов или блоков кода, которые больше не нужны.

Программный код до рефакторинга

В исходном коде mainForm.cs часть функционала находилась прямо внутри обработчика событий, например, загрузка данных для разных вкладок (Request, Comment, Equipment, и др.) или различие интерфейса по ролям пользователя. Фрагмент до рефакторинга представлен на листинге 3.1.

Листинг 3.1 – Программный код до рефакторинга:

private void mainForm\_Load(object sender, EventArgs e)

{

loadRequest();

differentationPole();

}

Программный код после рефакторинга

Выделение методов загрузки данных по вкладкам

Для каждой вкладки создан отдельный метод: loadRequest(), loadComment(), loadEquipment(), loadClient(), loadSpecialist(), loadReport(). Теперь обработчик событий просто вызывает соответствующие методы. Фрагмент после рефакторинга представлен на листинге 3.2.

Листинг 3.2 – Программный код после рефакторинга:

private void mainForm\_Load(object sender, EventArgs e)

{

loadRequest();

differentationPole();

}

Выделение метода настройки ролей пользователя

Логика различия доступных функций для ролей (client, specialist, admin) вынесена в отдельный метод differentiationRole() вместо того, чтобы дублировать эту логику в разных местах.

Выделение общих операций с таблицами

Методы cleanTable(DataTable table) и configureDGV(DataGridView dgv) реализуют обработку и настройку отображения данных – вместо того, чтобы копировать этот код в каждый метод загрузки данных.

**3.2 Разработка тестов. Отладка и тестирование программы на уровне модуля**

Отладка и тестирование программы на уровне модуля — это процесс проверки и исправления функциональности отдельных, изолированных компонентов (модулей, функций, классов) программного обеспечения.

Особенности тестирования программ

* тестирование проводится для проверки корректности функционирования программы в различных условиях;
* в процессе тестирования выявляются дефекты, которые могут быть связаны с логикой, интерфейсом, производительностью и другими аспектами приложения;
* тестирование может быть автоматическим и ручным.

Виды тестирования

* модульное (unit-тестирование): проверка отдельных функций/методов;
* интеграционное: проверка совместной работы нескольких модулей;
* системное: тестирование всего приложения целиком;
* приёмочное: проверка соответствия приложения требованиям заказчика.

Методы тестирования

* чёрный ящик: тестировщик не знает внутреннего устройства программы, только входные и выходные данные;
* белый ящик: тестировщик знает код и тестирует все возможные ветвления;
* серый ящик: комбинированный подход.

Методы отладки

* установка точек останова (breakpoints);
* пошаговое выполнение (Step Into, Step Over, Step Out);
* просмотр текущих значений переменных и состояния программы;
* анализ стека вызовов (Call Stack);
* использование окон Watch, Locals, Autos для анализа переменных.

В среде Visual Studio открыт файл mainForm.cs.

Использование команд пошагового выполнения кода

Запущена программа.

При достижении точки останова использовала команды:

* Step Into (F11): для входа внутрь метода (loadRequest()).
* Step Over (F10): для перехода к следующей строке текущего метода.
* Step Out (Shift+F11): для выхода из текущего метода к вызвавшему.

Проверка переменных

В окне отладчика видны значения локальных переменных.

Навела курсор на переменную query, чтобы посмотреть текущее значение.

**3.3 Анализ результатов тестирования. Тестирование с помощью инструментов среды разработки**

Методика анализа результатов тестирования

Анализ результатов тестирования – это процесс оценки правильности работы программы на основе сравнения фактических и ожидаемых результатов тестов.

Инструменты тестирования программного обеспечения

В современной практике активно применяются:

* фреймворки модульного тестирования.
* обозреватель тестов – инструмент Visual Studio для запуска, просмотра и анализа результатов тестов.
* анализ покрытия кода – показывает, насколько полно код проверяется тестами.

Создание проекта модульных тестов

В среде Visual Studio был создан отдельный проект типа Unit Test Project MSTest, подключён к основному проекту приложения.

Реализация модульных тестов

Для методов приложения были реализованы модульные тесты.

Модульный тест: CleanTable\_ShouldTrimAllStringValues

Цель теста: проверить, что метод cleanTable корректно очищает все текстовые значения в таблице от лишних символов.

Модульный тест: ConfigureDGV\_ShouldSetGridProperties

Цель теста: убедиться, что метод configureDGV корректно настраивает свойства визуализации таблицы DataGridView для единого стиля оформления.

Модульный тест: DifferentiationRole\_ClientRole\_ConfiguresButtonsCorrectly

Цель теста: проверить, что метод differentiationRole правильно настраивает видимость кнопок для пользователя с ролью «client».

Запуск и анализ тестов через обозреватель тестов

В меню Visual Studio был открыт Test Explorer (Обозреватель тестов). Все реализованные тесты были запущены командой Run All.

**4 Задание 4 Разработка мобильного приложения**

**4.1 Создание интерфейса мобильного приложения**

Activity — это компонент приложения, который служит точкой входа для взаимодействия приложения с пользователем, а также отвечает за то, как пользователь перемещается внутри приложения или между приложениями.

Файлы разметки графического интерфейса для Activity в проекте располагаются в каталоге res/layout. По умолчанию при создании проекта с пустой активностью уже есть один такой файл — activity\_main.xml.

В этом файле определяются все графические элементы и их атрибуты, которые составляют интерфейс. Например, можно указать размеры, цвета, свойства элементов.

Чтобы установить разметку для текущей активности, в методе Activity.onCreate нужно передать ссылку на ресурс разметки в метод setContentView(). Название ресурса (layout) совпадает с именем файла.

ConstraintLayout — менеджер макетов для Android, который позволяет создавать гибкие и адаптивные пользовательские интерфейсы. Он позволяет определять положение и размер элементов интерфейса с помощью ограничений (constraints), которые задают отношения между элементами и макетом.

Процесс построения UI из пользовательского макета включает следующие этапы:

* Исследование. Сбор и анализ информации о целевой аудитории, потребностях и предпочтениях пользователей.
* Прототипирование. Создание первоначальных эскизов и макетов, которые помогают визуализировать идеи и концепции.
* Дизайн. Создание визуального стиля и графических элементов интерфейса: выбор цветовой схемы, разработка типографики, создание иконок и графики.
* Тестирование. Проверка готового интерфейса на различных устройствах, выявление и исправление ошибок.
* Реализация. Дизайнеры и разработчики совместно работают над внедрением интерфейса в продукт: создают CSS-стили, HTML-код, JavaScript-компоненты, интегрируют интерфейс с системой.

**Постановка задачи:**

Создать мобильное приложение с помощью Android Studio в соответствии с техническим заданием задания 2.1.

**Выполнение:**

Я создала проект в Android Studio. На форму добавила элементы: TextView, EditText, Button. Таким образом, я создала форму авторизации. Получившийся интерфейс формы регистрации представлен на рисунке 4.1.



Рисунок 4.1 – Форма авторизации

Программный код формы авторизации представлен в листинге 4.1.

Листинг 4.1 – Код для формы авторизации

public class MainActivity extends AppCompatActivity {

private EditText editTextLogin;

private EditText editTextPassword;

private Button buttonLogin;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

editTextLogin = findViewById(R.id.editTextText);

editTextPassword = findViewById(R.id.editTextText2);

buttonLogin = findViewById(R.id.button4);

buttonLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String login = editTextLogin.getText().toString();

String password = editTextPassword.getText().toString();

if (login.equals("admin") && password.equals("1234")) {

Intent intent = new Intent(MainActivity.this, MainActivity2.class);

startActivity(intent);

} else {

Toast.makeText(MainActivity.this, "Неверный логин или пароль", Toast.LENGTH\_SHORT).show();

}

}

});

}

Создала главную форму, она нужна для открытия таблиц по нажатию на нужную кнопку.

Интерфейс формы представлен на рисунке 4.2.

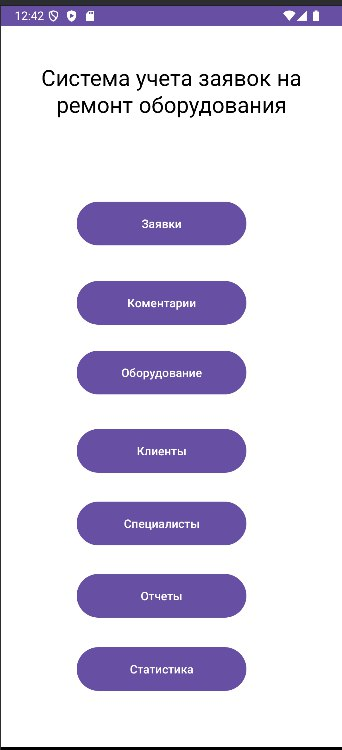


Рисунок 4.2 – Главная форма

Программный код для главной формы представлен в листинге 4.2.

Листинг 4.2 – Код для главной формы

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main2);

button1= findViewById(R.id.button1);

button2= findViewById(R.id.button2); button3= findViewById(R.id.button3);

button4= findViewById(R.id.button4); button5= findViewById(R.id.button5);

button6= findViewById(R.id.button6); button7= findViewById(R.id.button7);

button1.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity3.class);

startActivity(intent); }

}); button2.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity4.class);

startActivity(intent); }

}); button3.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity5.class);

startActivity(intent); }

}); button4.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity6.class);

startActivity(intent); }

}); button5.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity7.class);

startActivity(intent); }

}); button6.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity8.class);

startActivity(intent); }

}); button7.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intent = new Intent(MainActivity2.this, MainActivity9.class);

startActivity(intent); }

});}

}

Далее нажимая на выбранную кнопку открывается нужная таблица, где есть возможность фильтрации по значению, а также добавления, изменения и удаления данных. Предоставляется кнопка «Назад».

Интерфейсы форм «Заявки», «Оборудование» и «Клиенты» представлены на рисунках 4.3.1, 4.3.2, 4.3.3. Данный интерфейс аналогичен для других форм с таблицами.

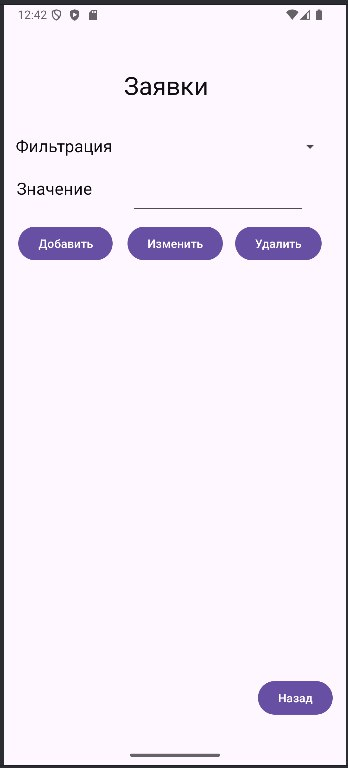


Рисунок 4.3.1 – Форма «Заявки»

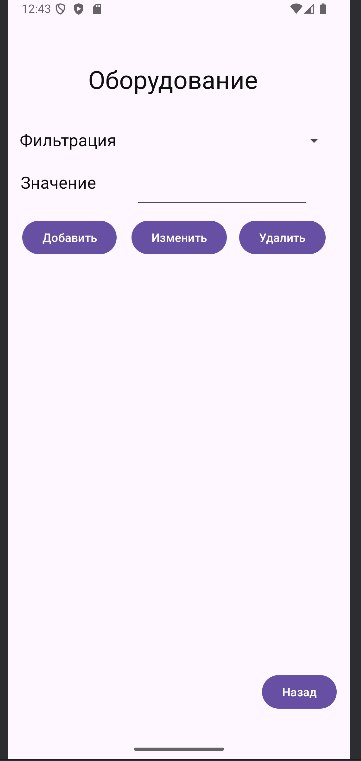


Рисунок 4.3.2 – Форма «Оборудование»

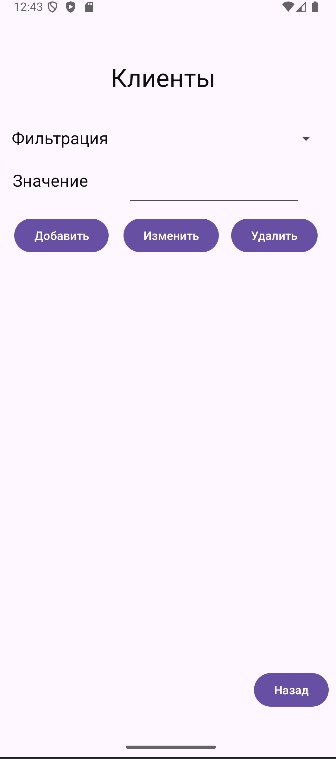


Рисунок 4.3.3 – Форма «Клиенты»

Программный код для формы «Заявки» представлен в листинге 4.3. Для остальных форм с таблицами код аналогичен.

Листинг 4.3 – Код для формы «Заявки»

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

EdgeToEdge.enable(this); setContentView(R.layout.activity\_main3);

button10 = findViewById(R.id.button10);button9 = findViewById(R.id.button9);

button = findViewById(R.id.button);button8 = findViewById(R.id.button8);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> { Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom); return insets;

});

button10.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

Intent intent = new Intent(com.example.myapplication.MainActivity3.this, MainActivity2.class); startActivity(intent);

} });

}

Также есть возможность перейти на форму статистика, на которой представлены выполненные заявки и среднее время выполнения заявок. Также предусмотрена возможность вернуться назад по нажатию на кнопку.

Интерфейс представлен на рисунке 4.4.

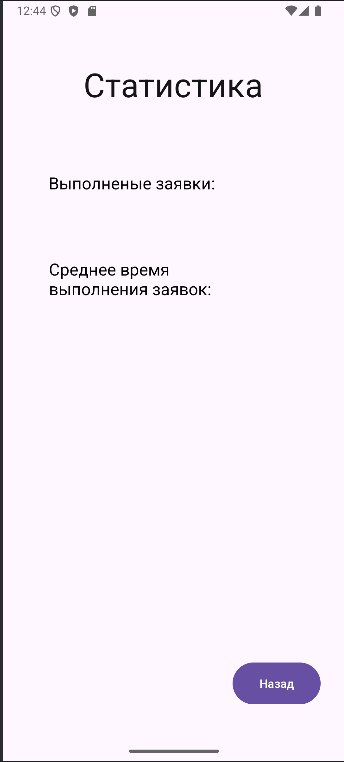


Рисунок 4.4 – Форма статистики

**4.2 Тестирование и оптимизация мобильного приложения**

Мобильное тестирование — это процесс проверки программного обеспечения, разработанного для мобильных устройств (смартфонов, планшетов), на соответствие критериям качества, функциональности и производительности.

Автоматизированное тестирование — это проверка ПО с использованием специальных программных инструментов, которые выполняют тесты автоматически, без участия человека. Тестировщик создаёт скрипты или сценарии тестирования, которые содержат инструкции для выполнения определённых действий и проверки результатов.

Ручное тестирование — это проверка ПО вручную, без использования автоматизированных инструментов. Тестировщик взаимодействует с программой как обычный пользователь: проверяет функциональность, исследует интерфейс, оценивает удобство использования и выявляет потенциальные проблемы.

Этапы тестирования Android-приложений включают:

* Планирование. Определение целей, выбор инструментов, разработка стратегии тестирования и создание плана тестов.
* Настройка тестового окружения. Подбор тестовых девайсов, настройка эмуляторов и, возможно, облачных девайсов.
* Написание тест-кейсов. Создание сценариев (кейсов) тестирования на основе требований к продукту. При этом нужно учитывать функциональные и нефункциональные аспекты работы приложения.
* Выполнение тестов. Тестировщики осуществляют ручное тестирование приложений и автоматизированное, выявляют сбои и дефекты, фиксируют их и передают разработчикам для исправления.
* Тестирование производительности и безопасности. Проверка, как приложение ведёт себя при пиковых нагрузках, насколько быстро отвечает на запросы и защищено ли от потенциальных угроз.
* Создание отчётов. Создание итогового отчёта по результатам тестирования. Найденные и исправленные баги анализируются, определяются самые проблемные модули в приложении.
* Завершение тестирования. После завершения тестов и устранения всех выявленных проблем подготавливается отчёт, содержащий сводку тестовых действий, результаты, обнаруженные проблемы.

Тестирование Android-приложений имеет свои особенности, связанные с разнообразием устройств, спецификой пользовательского интерфейса и другими факторами.

Инструменты для ручного тестирования:

* Эмуляторы. Например, Android Studio — эмулятор, который поддерживает множество устройств.
* Инструменты для анализа сетевого трафика. Например, Charles Proxy — перехват и анализ HTTP/HTTPS-запросов.
* ADB (Android Debug Bridge). Консольный инструмент, позволяет устанавливать APK, делать скриншоты, логировать баги.

**Постановка задачи:**

Протестировать свое мобильное приложение из задания 2.1 любым инструментом.

**Выполнение:**

Для приложения, созданного на Android Studio был создан набор unit-тестов путём создания нового активити. В данном активити был написан набор тестов, который ввод Логин и Пароль в соответствующие поля, переходит на вторую активити, после чего на второй активити он нажимает на кнопку для перехода на активити с Таблицей «Клиенты».

**5 Задание 5 Задание работодателя**

Было выдано задание: сверстать главную страницу сайта ООО «АИС Город» по макету в приложении Figma. Для выполнения данного задания я использовала средства разработки Visual Studio Code и Figma для написания кода и использования макета.

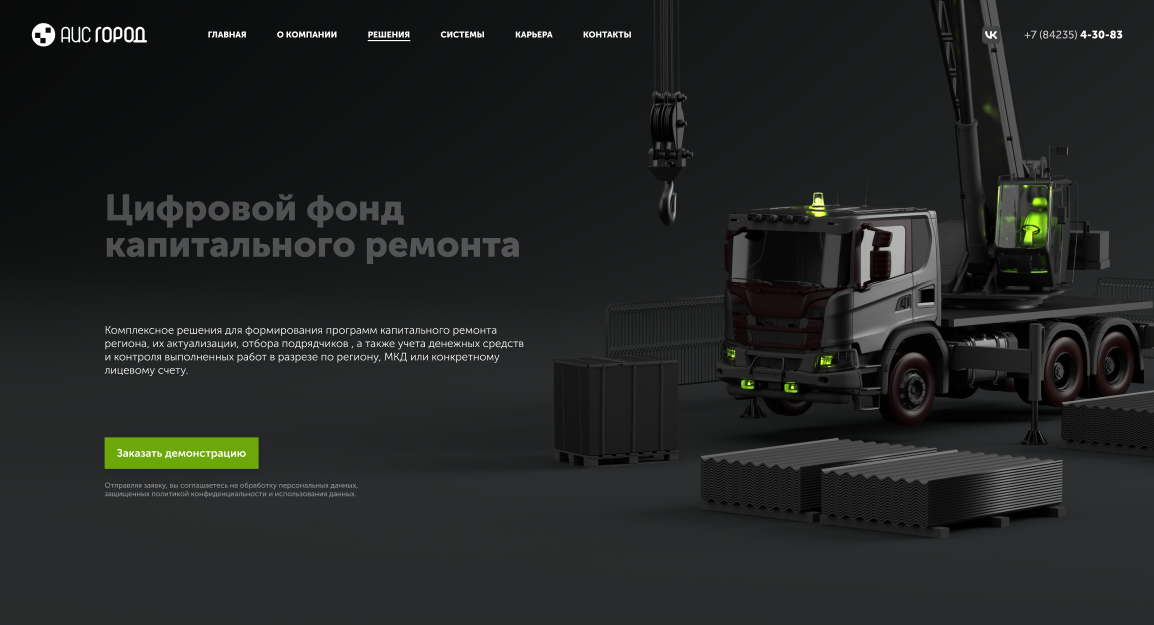


Рисунок 5.1 – Шапка c макета Figma

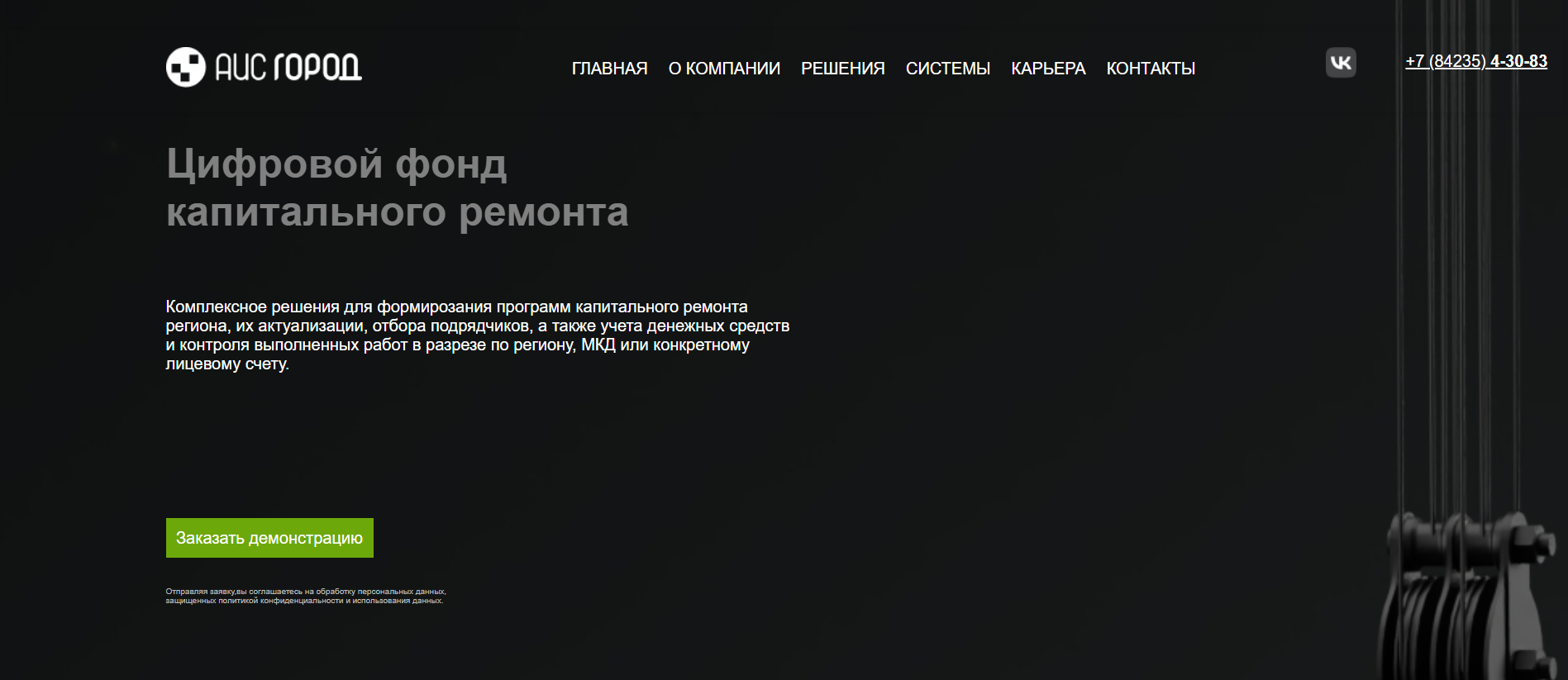


Рисунок 5.2 – Шапка макета в моем исполнении

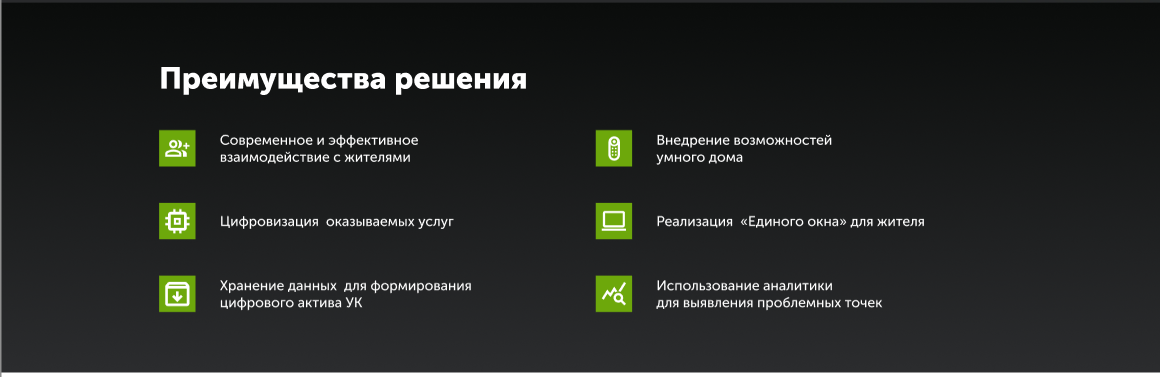


Рисунок 5.3 – «Преимущества решения» с макета Figma

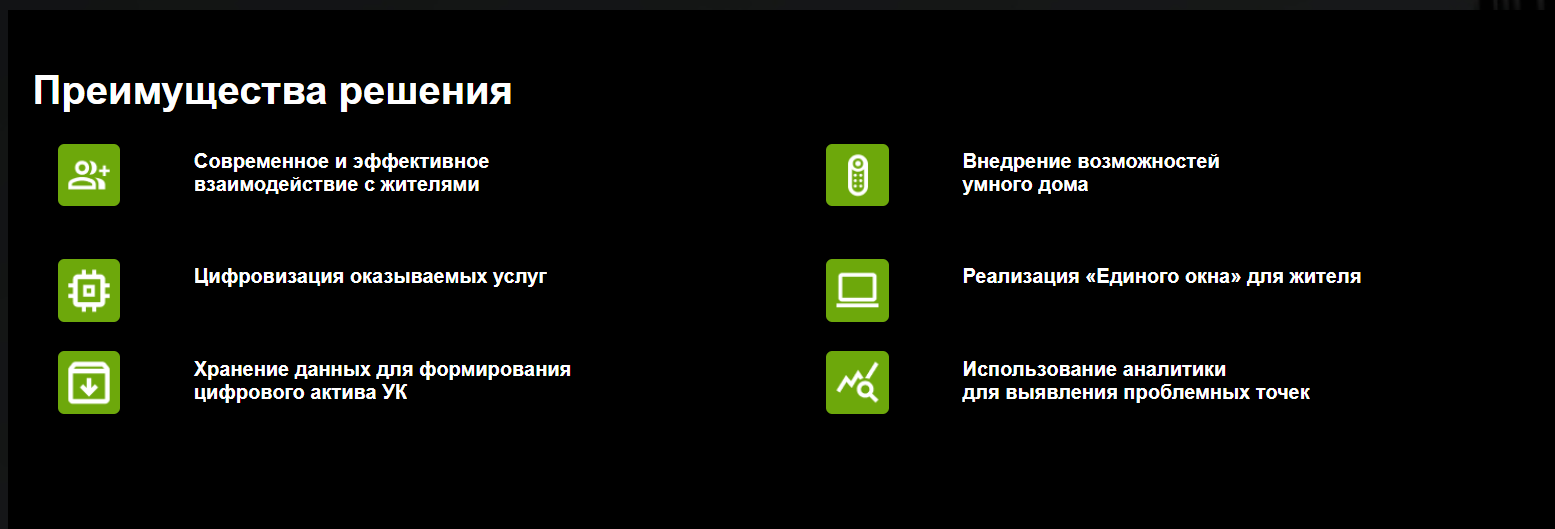


Рисунок 5.4 – «Преимущества решения» в моем исполнении

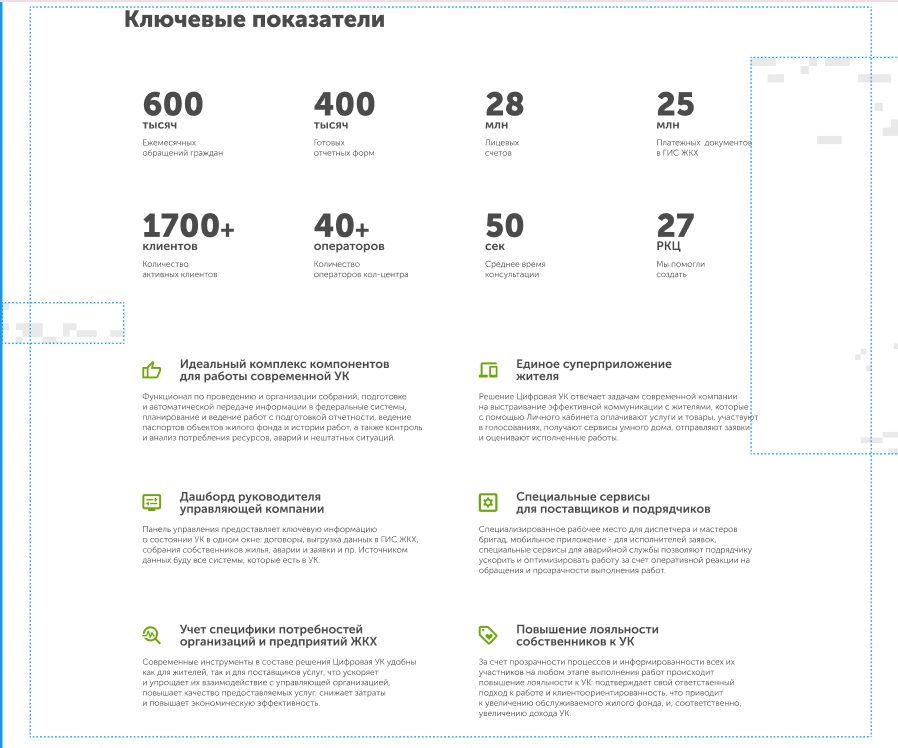


Рисунок 5.5 – «Ключевые показатели» с макета Figma



Рисунок 5.6.1 – «Ключевые показатели» в моем исполнении



Рисунок 5.6.2 – «Ключевые показатели» в моем исполнении

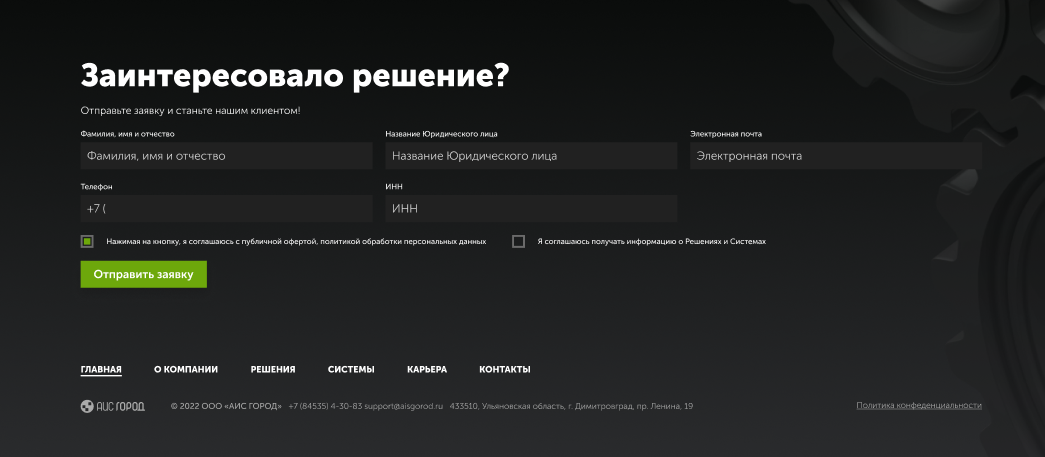


Рисунок 5.7 – «Подача заявки» с макета Figma

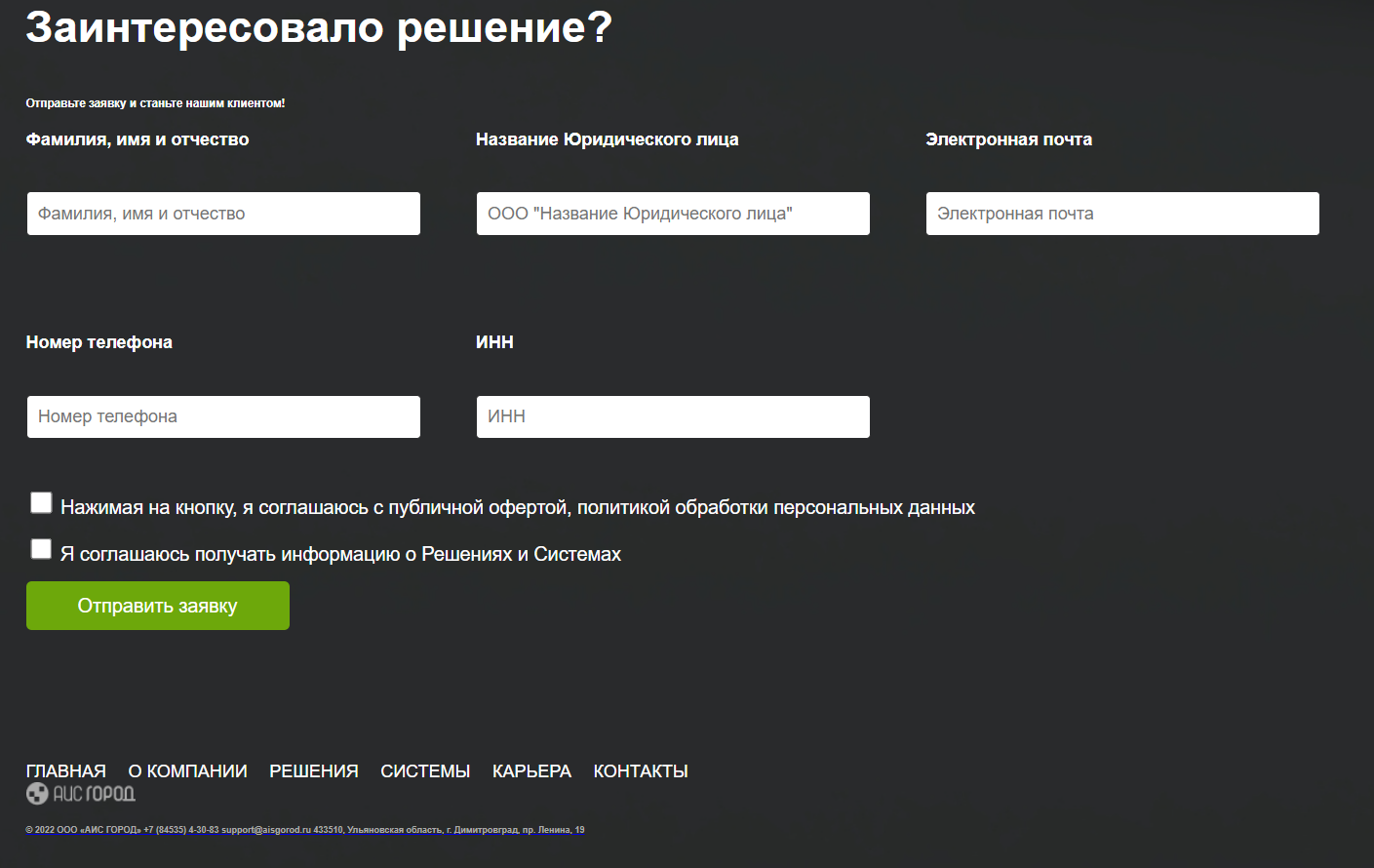


Рисунок 5.8 – «Подача заявки» в моем исполнении

**6 Задание 6 Оформление документации. Разработка презентации**

Задание:

1. Оформила отчет о прохождении производственной практики
2. Оформила презентацию.

Состав презентации:

1. Титульный лист
2. Содержание
3. О каждом задании кратко (на слайдах графический материал: скрины приложения/проекта/компиляция, таблицы и т.д.)
4. Заключение по производственной практике

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**1 Нормативно-правовые акты**

1. Федеральный закон от 26 июля 2017 г. № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации».

2. Федеральный закон от 27 июля 2006 г. № 152-ФЗ «О персональных данных».

3. Федеральный закон от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации».

**2 Учебная и специальная литература**

4. Морозов Е.А. Разработка программного обеспечения: методологии и инструменты. СПб.: БХВ-Петербург, 2023.

5. Кравцов С.В. Базы данных и СУБД. М.: ДМК Пресс, 2022.

6. Воронин И.Н. Информационная безопасность и защита данных. СПб.: Питер, 2024.

**3 Электронные ресурсы**

**7**. metanit.com посвящён различным языкам и технологиям программирования, компьютерам, мобильным платформам и ИТ-технологиям. https://metanit.com/

8. Сайт developer.android.com предназначен для разработчиков Android. https://developer.android.com/studio/write/add-resources?hl=ru

9. Сайт developer.mozilla.org (MDN Web Docs) — ресурс для веб-разработчиков, который содержит документацию, руководства и обучающие материалы по различным веб-технологиям, включая HTML, CSS и JavaScript. https://developer.mozilla.org/ru/

**Заключение  
о прохождении производственной практики**

Обучающаяся Степанова Дарья Дмитриевна 3 курса группы 331 специальности 09.02.07 Информационные системы и программирование проходила производственную практику по профессиональному модулю ПМ.01 Разработка модулей программного обеспечения для компьютерных систем с 12.05.2025 г. по 17.05.2025 г., с 26.05.2025 г. по 07.06.2025 г. в ООО «АИС Город».

По итогам производственной практики обучающаяся представила в ДИТИ НИЯУ МИФИ следующие документы:

1. Дневник по производственной практике (включая аттестационный лист и характеристику).

2. Отчёт по производственной практике.

Отчёт по производственной практике соответствует следующим характеристикам (нужное подчеркнуть):

- представлен своевременно/ несвоевременно;

- правильно и в полном объёме выполнены задания/ задания выполнены с ошибками/ задания не выполнены;

- приложения представлены в полном объеме/ приложения представлены в неполном объеме/ приложения не представлены.

Оформление дневника, отчета и приложений (нужное подчеркнуть):

- соответствует оформление дневника / не соответствует оформление дневника;

- соответствует оформление отчёта/ не соответствует оформление отчёта.

Результаты защиты отчёта производственнойпрактике, замечания и предложения:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Заключение о прохождении производственной практики по профессиональному модулю ПМ.01 Разработка модулей программного обеспечения для компьютерных систем:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка прописью)

**Руководитель практики от**

**образовательной организации \_\_\_\_\_\_\_\_\_** В.А. Романов

подпись

07.06.2025