МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство «Организация монтажных работ – "Бюро Монтажника"»

Исполнитель

студентка 2 курса группы 4 Зинович Елизавета Игоревна

Руководитель работы ассистент Северинчик Н.А.

Курсовой проект защищён с оценкой

Председатель Северинчик Н.А.

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc73122866)

[1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству 4](#_Toc73122867)

[2 Анализ требований к программному средству и разработка функциональных требований 6](#_Toc73122868)

[3 Проектирование программного средства 9](#_Toc73122869)

[3.1 Проектирование архитектуры приложения 9](#_Toc73122870)

[3.2 Проектирование логической структуры базы данных 11](#_Toc73122871)

[3.3 Проектирование логики сценариев использования 12](#_Toc73122872)

[4 Создание (реализация) программного средства 14](#_Toc73122873)

[4.1 Реализация сущностей 14](#_Toc73122874)

[4.2 Реализация архитектуры MVVM 15](#_Toc73122875)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 17](#_Toc73122876)

[6 Методика использования программного средства 20](#_Toc73122877)

[ЗАКЛЮЧЕНИЕ 26](#_Toc73122878)

[Список использованных источников 27](#_Toc73122879)

[Приложение А 28](#_Toc73122880)

[Приложение Б 29](#_Toc73122881)

[Приложение В 30](#_Toc73122882)

[Приложение Г 31](#_Toc73122883)

[Приложение Д 32](#_Toc73122884)

[Приложение Е 33](#_Toc73122885)

[Приложение Ж 34](#_Toc73122886)

# **ВВЕДЕНИЕ**

Курсовой проект посвящён разработке программного средства «Организация монтажных работ – “Бюро Монтажника”», основной целью которого является автоматизация работы компании, занимающейся выполнением монтажных работ в сфере отопительного оборудования.

В настоящее время, вероятно, вовсе не существует аналогов, реализованных в качестве настольных приложений. Поэтому разработка подобного программного средства может иметь практическую значимость для компаний, сферой деятельности которых являются монтажные работы.

Программное средство предусматривает три типа пользователей: клиент, бригада и администратор.

Интерфейс клиента содержит одно окно и позволяет оставить заявку на монтаж, а также найти заявки по данным клиента, соответственно получить актуальную информацию о статусе заявки.

Учитывая, что в сфере монтажных работ отопительного оборудования маловероятно наличие постоянных клиентов, роль клиента не предусматривает ни регистрации, ни авторизации. Это также позволяет клиенту максимально быстро оставлять заявку на монтаж. Гораздо более полезной окажется возможность, в случае указания актуального адреса электронной почты клиентом, получать с корпоративного аккаунта почты компании письма, уведомляющие о подтверждении заявки или о смене её статуса.

Интерфейс бригады представлен одним окном, доступ в которое предоставляется путем входа в аккаунт бригады под актуальными логином и паролем, и предусматривает он возможности просмотра назначенных ей заявок, просмотра информации о бригаде (а именно фамилий входящих в бригаду работников, а также общего количества заявок бригады) и смены статуса заявки как с уведомлением клиента, так и без него.

Наибольшее количество функций программного средства поддерживает интерфейс администратора, доступ к которому можно получить, совершив вход в аккаунт с логином «admin» и паролем «admin».

# **Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству**

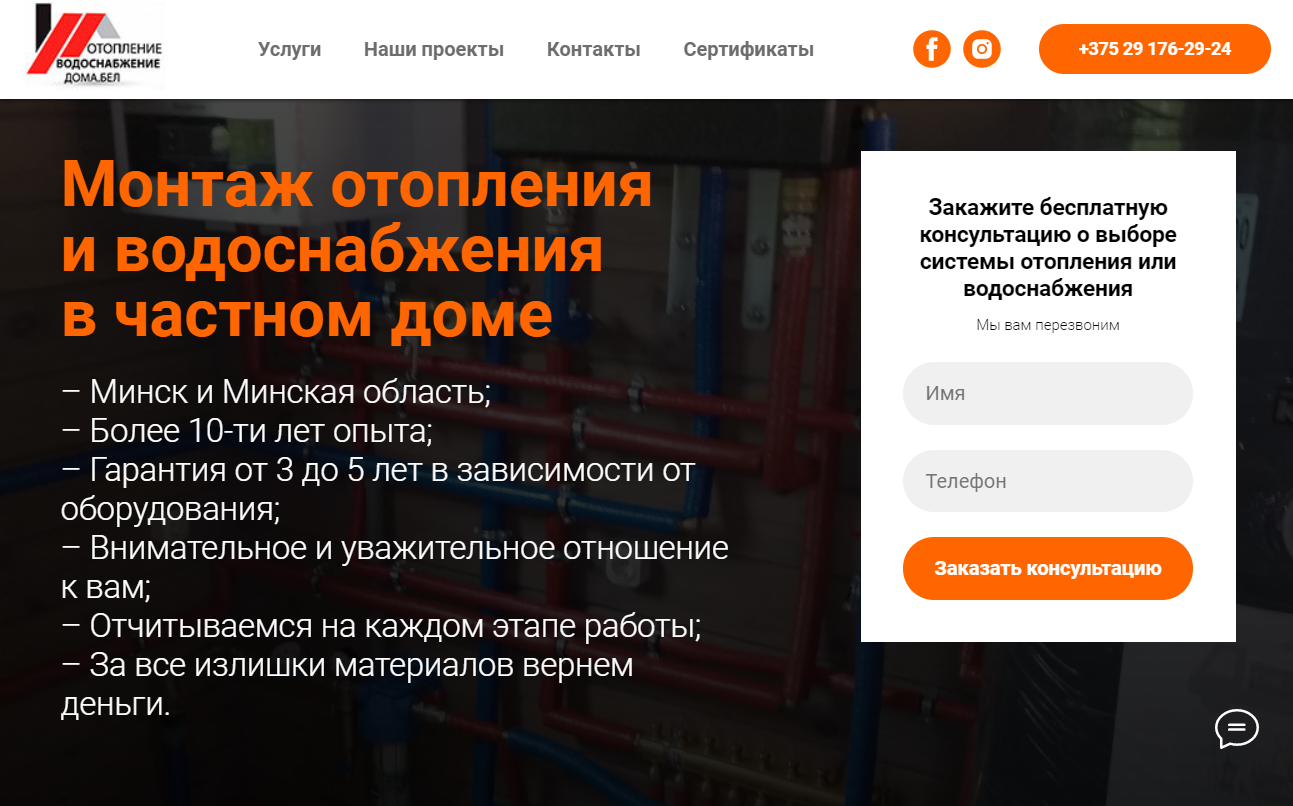
Вероятно, все существующие сервисы для организации заявок на монтаж представлены web-сайтами. Они имеют довольно ограниченный функционал для клиента, предоставляя возможность лишь оставить контактный номер телефона клиента для связи с сотрудником для консультации (рисунок 1.1).

Рисунок 1.1 Малоинформативная форма заявки на монтаж.

Существенным минусом такого клиентского интерфейса является отсутствие возможности указать в форме необходимые для обработки заявки данные, такие, как адрес или оборудование.

Так, форма на рисунке 1.2, будет более информативной, чем рассмотренная выше, так как содержит адрес объекта, комментарий к заявке. Такая информация, к тому же, является абстрагированной от технической, поэтому доступна к заполнению для любого клиента.

Однако и в этой форме не хватает некоторых важнейших данных, а именно монтируемого оборудования и стадии отделки (черновая, чистовая). Эта информация влияет на цену, сроки исполнения работ, на необходимые для выполнения заявки сотрудниками монтажной бригады инструменты и комплектующие – то есть, является основополагающей при автоматизации процесса обработки подобных заявок.

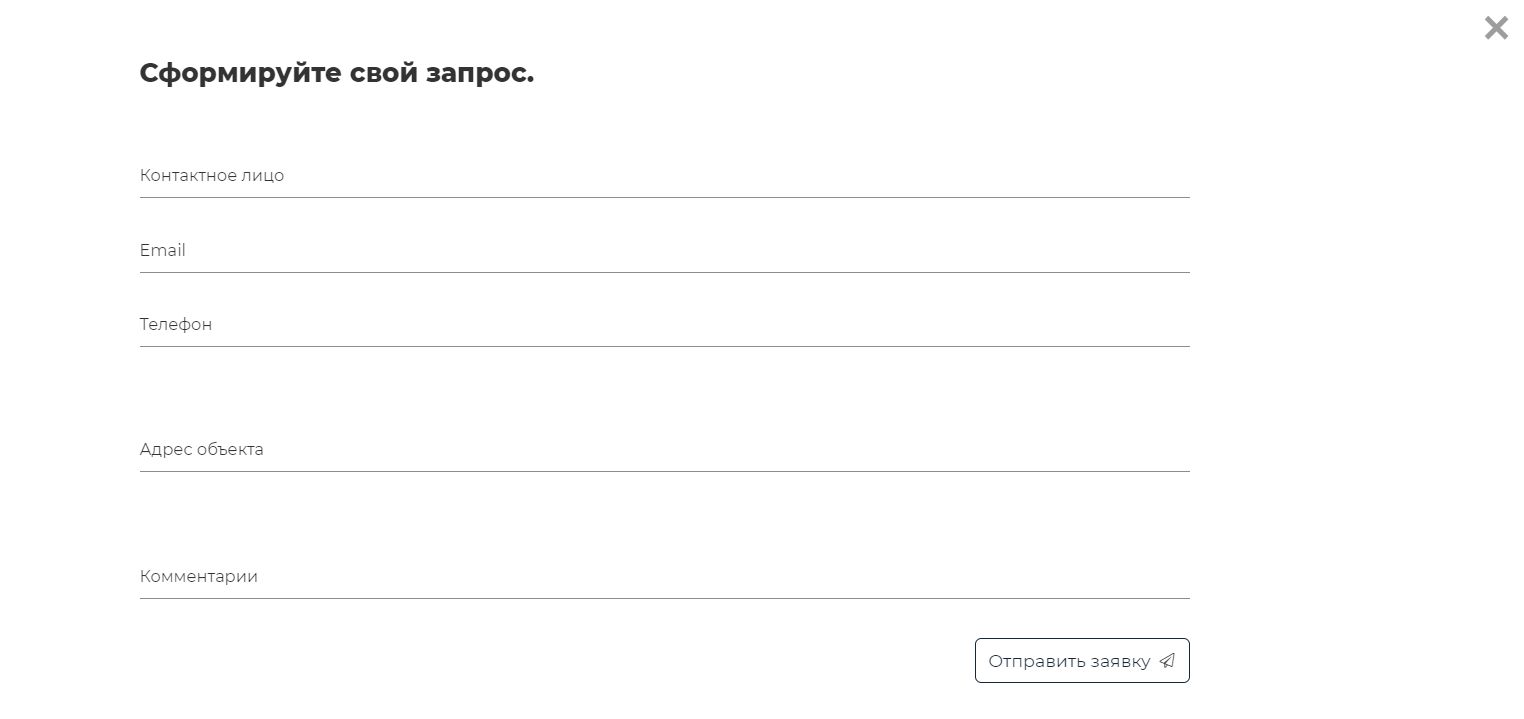


Рисунок 1.2 Практически полная форма заявки на монтаж.

Поэтому к проектируемому программному средству следует предъявить следующие требования.

Форма заявки клиента должна содержать:

* поля для ввода личной информации о контактном лице (заказчике работ);
* поля для указания адреса выполнения работ;
* возможность выбрать количество монтируемого оборудования;
* указание стадий отделки;
* возможность выбрать дату выполнения работ.

Полностью заполненная форма такого образца позволит компании не держать в штате сотрудников-менеджеров, предоставляющих каждому клиенту примерно одинаковую информацию при консультации по телефону, а сэкономить компании как время, так и деньги, автоматизировав процесс оставления и обработки заявки.

Значимым преимуществом должно быть подтверждение заявки, приходящее на указанный адрес электронной почты.

Для обработки подобной формы заявки потребуется администратор, непосредственно бригада-исполнитель, а также предварительно внесенные в базу данные о стадиях отделки, необходимых инструментах и типах оборудования.

Для удовлетворения программным средством поставленных требований возникает необходимость использования дополнительной литературы.

Главным источником информации станет официальная документация языка С# — https://docs.microsoft.com/en-us/dotnet/csharp/.

Также применима будет информация с сайтов METANIT, раздела C# (<https://metanit.com/sharp/>) и WPF Tutorial(https://wpf-tutorial.com).

# **Анализ требований к программному средству и разработка функциональных требований**

Целью курсового проекта является разработка программного средства, предназначенного для автоматизации процесса обработки заявок на монтаж, а также для выполнения ряда дополнительных функций: добавления бригад, редактирования клиентов, заявок и сотрудников, поиска по заявкам и фильтрации.

При разработке программного средства будут задействованы:

* язык программирования C#;
* технология WPF;
* язык разметки XAML;
* СУБД Microsoft SQL Server;
* технология Entity Framework;
* библиотеки Material Design (Themes, Colors);
* библиотека LiveCharts.

Программный продукт должен быть реализован на объектно-ориентированном языке программирования C# с использованием технологии WPF.

Технология WPF (Windows Presentation Foundation) является частью экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов, использующую язык XAML.

XAML (eXtensible Application Markup Language) - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF язык используется прежде всего для создания пользовательского интерфейса декларативным путем.

Все данные программного средства должны храниться в базе данных. Она разрабатывается с использованием системы управления базами данных (СУБД) Microsoft SQL Server.

SQL Server характеризуется такими особенностями как:

* производительность (работает очень быстро);
* надежность и безопасность;
* простота (с данной СУБД относительно легко работать и вести администрирование).

Для упрощения работы с базой данных также применима технология Entity Framework (EF), которая позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. EF — это рекомендованная Microsoft технология доступа к данным для новых приложений.

В EF существует несколько подходов к работе, или по-другому, несколько способов создания этой модели:

* Database First – создание модели сущностей и объектной модели (классы) из существующей базы данных. Описание модели сущностей сохраняется в виде xml файла, по которому EF создает модель EDM;
* Model First — создание xml-файла модели при помощи дизайнера, на основании которой генерируются база данных и классы (объектная модель);
* Code First – написание кода классов при отсутствии модели и БД. Далее на основе этих классов происходит генерация базы данных и модели сущностей EDM.

При разработке программного средства «Организация монтажных работ – “Бюро Монтажника”» будет применен подход Code first.

Библиотека Material Design – одна из самых популярных библиотек графического интерфейса для WPF, содержащая стили для большинства стандартных элементов управления, а также множество дополнительных элементов управления, с помощью которых можно создавать привлекательный дизайн.

Библиотека Live Charts предоставляет возможность визуализации данных в виде графиков и диаграмм. В программном средстве задействуется круговая диаграмма Pie Chart для отображения статистики заявок и диаграмма Heat Series для составления расписания бригад. Использование этой библиотеки для отображения данных позволит придерживаться шаблона MVVM, а также сохранять привлекательный дизайн программного средства.

Приложение должно поддерживать респонсивный дизайн в окне администратора.

При разработке программного средства «Организация монтажных работ – “Бюро Монтажника”» используется шаблон проектирования MVVM (который также подразумевает использование Command), а также паттерн Repository.

Программное средство должно иметь возможность запускаться без использования IDE (интегрированных сред разработки). Единовременно может быть запущен лишь один экземпляр приложения.

Также приложение должно предоставлять пользователям возможность входа с актуальными логином и паролем, производить валидацию вводимых пользователем данных, обрабатывать возникающие исключительные ситуации.

Чтобы соответствовать ранее выработанным требованиям, программное средство должно предоставлять пользователям, в зависимости от их роли, следующие функциональные возможности.

Функции клиента:

* оставить заявку на монтаж;
* получить актуальную информацию о статусе заявки;
* получить список всех своих заявок (как находящихся в процессе, так и готовых).

Функции бригады:

* выполнение авторизации: вход под логином и паролем, состоящим из слова «brigade» и номера бригады (например, для бригады 1022 и логин, и пароль будут brigade1022);
* просмотр своих заявок, информации о них;
* оповещение клиента при смене статуса заявки.

Функции администратора сервиса:

* выполнение авторизации (входа под логином admin и паролем admin);
* выполнение регистрации новых бригад (при создании новой бригады автоматически создаётся аккаунт для бригады, принцип формирования логина и пароля которого уже описаны выше в функциях бригады;
* обработка заявок от клиентов (назначение заявки конкретной бригаде, смена статуса заявки);
* оповещение клиента при смене статуса заявки;
* просмотр расписания бригад и управление им (в контексте назначения бригадам заявок).

Функциональные возможности отображены на UML-диаграмме вариантов использования, приведённой на рисунке 2.1 (Приложение А).

# **Проектирование программного средства**

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе пояснительной записки. Этап проектирования подразумевает их анализ.

## **3.1 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы.

Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон MVVM (Model-View-ViewModel), который позволяет отделить логику приложения от визуальной части (представления).

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

* модели описывают используемые в приложении данные (могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели, не могут содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления);
* представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением;
* модель представления связывает модель и представление через механизм привязки данных; если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны; также содержит логику по получению данных из модели, которые потом передаются в представление; определяет логику по обновлению данных в модели.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

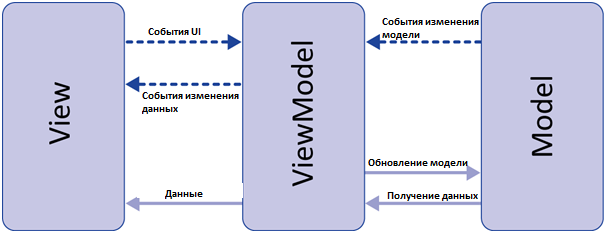


Рисунок 3.1 Структура шаблона MVVM.

Для грамотного проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. В приложении Б представлена диаграмма деятельности, описывающая процесс входа в приложение под разными ролями.

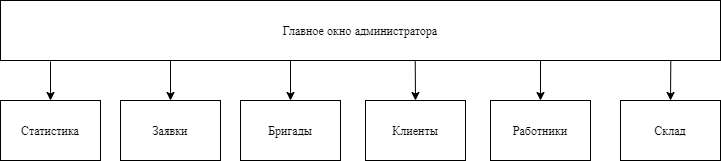
При запуске приложения появляется окно, предлагающее войти как клиент или перейти на страницу входа в аккаунт.

При выборе варианта «войти как клиент» открывается окно клиента. Если перейти на страницу входа в аккаунт, можно ввести в поля для логина и пароля данные, пройти валидацию и войти в приложение под соответствующей ролью. Причем, если роль вошедшего пользователя – администратор, произойдет вход в окно администратора, а если бригада – в окно бригады.

После входа в приложение под клиентом откроется окно, содержащее форму для заявки, строку поиска заявок и место для отображения перечня заявок клиента.

После входа в приложение под бригадой откроется окно, содержащее краткую информацию о бригаде, строку поиска заявок и место для отображения перечня заявок бригады.

После входа в приложение под администратором откроется окно с шестью страницами, по умолчанию активной будет «Статистика».

Рисунок 3.2 Схема главного окна администратора.

На странице «Статистика» будут представлена общая информация о количестве клиентов, заявок. Будет приведены диаграмма, отражающая отношение количества заявок по статусам готовности, а также отображено расписание на неделю для всех бригад.

На странице «Заявки» будет отображен список заявок, краткий перечень бригад, а также кнопки для редактирования заявки, для сокрытия готовых заявок, отображения всех заявок (в том числе и готовых) и для показа информации об оборудовании заявки.

На странице «Бригады» будет отображаться список с существующими бригадами. Будут предусмотрены возможности удаления и добавления новой бригады.

На странице «Клиенты» можно будет посмотреть весь список клиентов компании, об их заявках, а также редактировать или удалить клиента, осуществить поиск.

На странице «Работники» можно будет посмотреть весь список работников компании, а также добавить, редактировать или удалить работника, определить в бригаду, осуществить поиск.

На странице «Склад» отображен список инструментов, комплектующих и приборов для монтажа по типам, необходимых при обработке заявки.

Также, независимо от активной страницы, на верхней панели окна администратора необходимы кнопки для отображения настроек аккаунта корпоративной почты, для выхода из окна и перехода в окно входа в приложение.

## **3.2 Проектирование логической структуры базы данных**

Для хранения данных программного средства «Организация монтажных работ – “Бюро Монтажника”» была создана база данных TheBuerau. Для этого использовалась система управления реляционными базами данных Microsoft SQL Server 2019.

База данных состоит из 12 таблиц, и ее схема приведена на рисунке в приложении В.

Ниже – краткое описания назначения таблиц базы данных TheBuerau.

Таблица Company содержит данные для входа в аккаунт корпоративной почты (логин, пароль).

Таблица Role содержит описания существующих ролей в приложении.

Таблица User представляет собой описание конкретного пользователя приложения, с указанием роли и данных для входа.

Таблица Brigade содержит бригаду с указанием на её аккаунт для входа в приложение.

Таблица Employee содержит информацию о работнике бригады.

Таблица Request служит для хранения заявки на монтаж.

Таблица Client хранит информацию о клиенте, сделавшем заявку.

Таблица Address хранит информацию об адресе, сделавшем заявку

Таблица RequestEquipment хранит информацию о монтируемом оборудовании для заказа.

Таблица Equipment содержит всё возможное оборудование.

Таблица Accessory содержит все возможные комплектующие для разных типов оборудования.

Таблица Tool содержит все возможные инструменты для выполнения монтажных работ на разных стадиях отделки.

## **3.3 Проектирование логики сценариев использования**

При проектировании приложения были рассмотрены различные сценарии использования. На рисунке 3.3 представлена диаграмма последовательности поиска клиентом своих заявок.

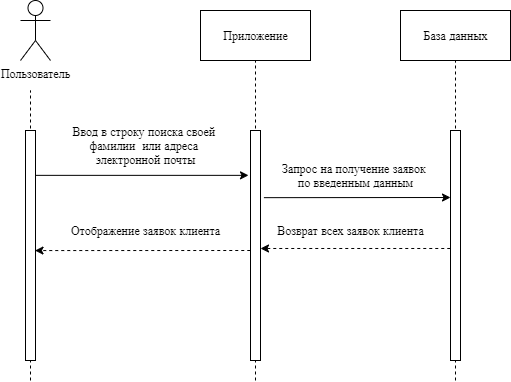


Рисунок 3.3 Диаграмма последовательности поиска клиентом своих заявок.

Изначально клиенту необходимо нажать ввести в строку поиска необходимые данные. По мере ввода данных приложение будет отправлять запрос в базу данных на получение соответствующих поисковому запросу заявок. Далее, когда база данных вернет информацию о найденных заявках, эта информация будет отображена в списке заявок. На этой странице присутствует кнопка добавления в избранное.

# **Создание (реализация) программного средства**

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

## **4.1 Реализация сущностей**

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Code first. При данном подходе модель EDMX не используется.

Создание базы данных происходит из созданной вручную модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.2.

Диаграмма классов UML для сущностных классов представлена на рисунке в приложении Г.

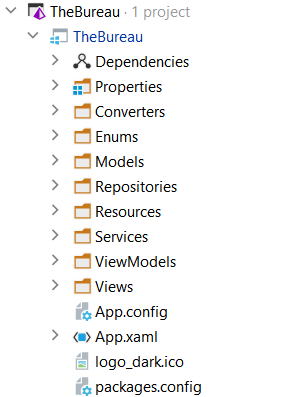
На диаграмме классов видно, как связаны между собой сущности.

Например, Сущность User связана с сущностями Brigade и Role отношениями композиции. То есть при уничтожении объекта Role вместе с ним будут уничтожены и User, связанные с ним, а при уничтожении User – связанные с ним объекты Brigade.

Таким же отношением связаны Request и RequestEquipment, а также Equipment с RequestEquipment и Accessory с Equipment.

Сущности Employee и Brigade связаны отношением ассоциации, так как в бригаду может входить несколько работников.

## **4.2 Реализация архитектуры MVVM**

Для реализации паттерна MVVM файлы программного средства были распределены по соответствующим директориям и реализовали соответствующие функции.

Разделение проекта на логические модули представлено на рисунке 4.1.

Рисунок 4.1 Логические модули проекта.

В папке Converters расположены классы конвертеры значений, которые позволяют преобразовать значение из источника привязки к типу, который понятен приемнику привязки. Эти классы реализуют интерфейс IValueConverter или IMultiValueConverter.

В папке Enums содержатся вспомогательные классы перечислений для описания возможных ролей, стадий отделки и статусов заявки.

В папке Models содержатся сущностные классы, которые используются для создания БД. Их описание представлено в разделе 4.1.

Для удобной работы с данными, используется паттерн Repository. Он позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

В папке Resources находятся папка Images, в которой находятся изображения, используемые приложением, а также файлы со стилями и цветами для приложения.

В папке Services содержатся классы, содержащие сообщения для ошибок валидации, отправки сообщений по электронной почте и хеширования паролей.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде класса RelayCommand код которого приведен в приложении Д.

Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

В папке ViewModels располжены все классы ViewModel для каждого представления. Также там содержится абстрактный класс ViewModelBase код которого предоставлен в приложении Е, и от которого наследуются все классы ViewModel. Класс реализует интерфейс INotifyPropertyChanged.

В папке Views хранятся все разработанные представления и находится папка с Controls, содержащая пользовательские элементы управления.

Для разработки графической части приложения была выбрана технология WPF.

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В конечном итоге в приложении были реализованы следующие окна:

* стартовое окно (вход как клиент и вход в аккаунт);
* главное окно клиента;
* главное окно бригады;
* главное окно администратора (содержит различные страницы);
* окно для редактирования настроек корпоративной почты;
* окно для редактирования клиента;
* окно для добавления работника;
* окно для редактирования работника;
* окно редактирования заявки от администратора;
* окно редактирования заявки от бригады.

А также следующие страницы:

* для входа как клиент (без авторизации);
* для входа как администратор или бригада (с авторизацией);
* страница со статистикой;
* страница бригад;
* страница заявок;
* страница клиентов;
* страница работников;
* страница склада;

Были разработаны пользовательские элементы управления для отображения заявки, клиента, бригады и некоторые другие.

В результате выполнения данного этапа было создано функционирующее программное средство.

# **Тестирование, проверка работоспособности и анализ полученных результатов**

Прежде всего были проведены тесты авторизации, такие как проверка на неверные данные.

Валидация выполнена таким образом, что кнопка для входа или регистрации будет недоступна до тех пор, пока введённые данные не будут корректными. При таком способе валидации шанс ввести недопустимые данные минимален. На рисунке 5.1 видно, что валидация на странице регистрации не допускает ввода некорректных полей.

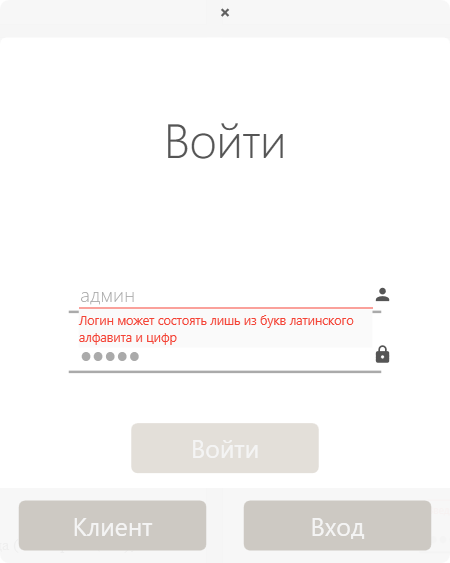
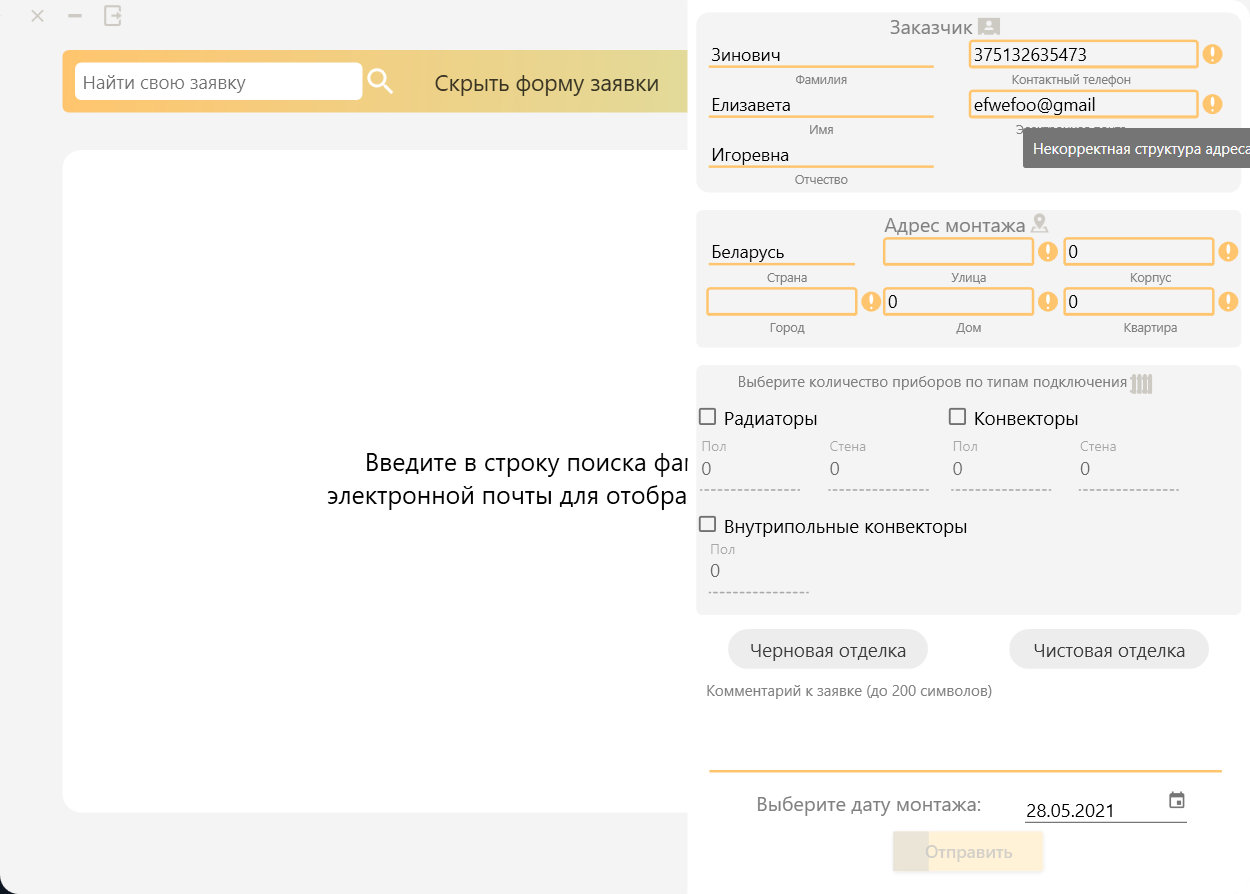
В поле логина можно ввести только буквы строчные и прописные латинского алфавита и цифры, длина логина не может превышать 20 символов. Пароль может иметь длину от 5 до 20 символов.

Рисунок 5.1 Ввод недопустимого логина – символы кириллического алфавита.

Также были проведены тесты на работу валидации внутри приложения. Наличие валидации является обязательным в связи с тем, что в приложении ведётся работа с базой данных.

В форме заявке также все поля подвергаются валидации. Например, нельзя ввести номер телефона, в котором код региона отличен от 17, 25 29, 33 или 44. При указании адреса электронной почте недопустимо отсутствие домена. Это отображено на рисунке 5.2.

Рисунок 5.2 Ввод недопустимых данных в форму заявки.

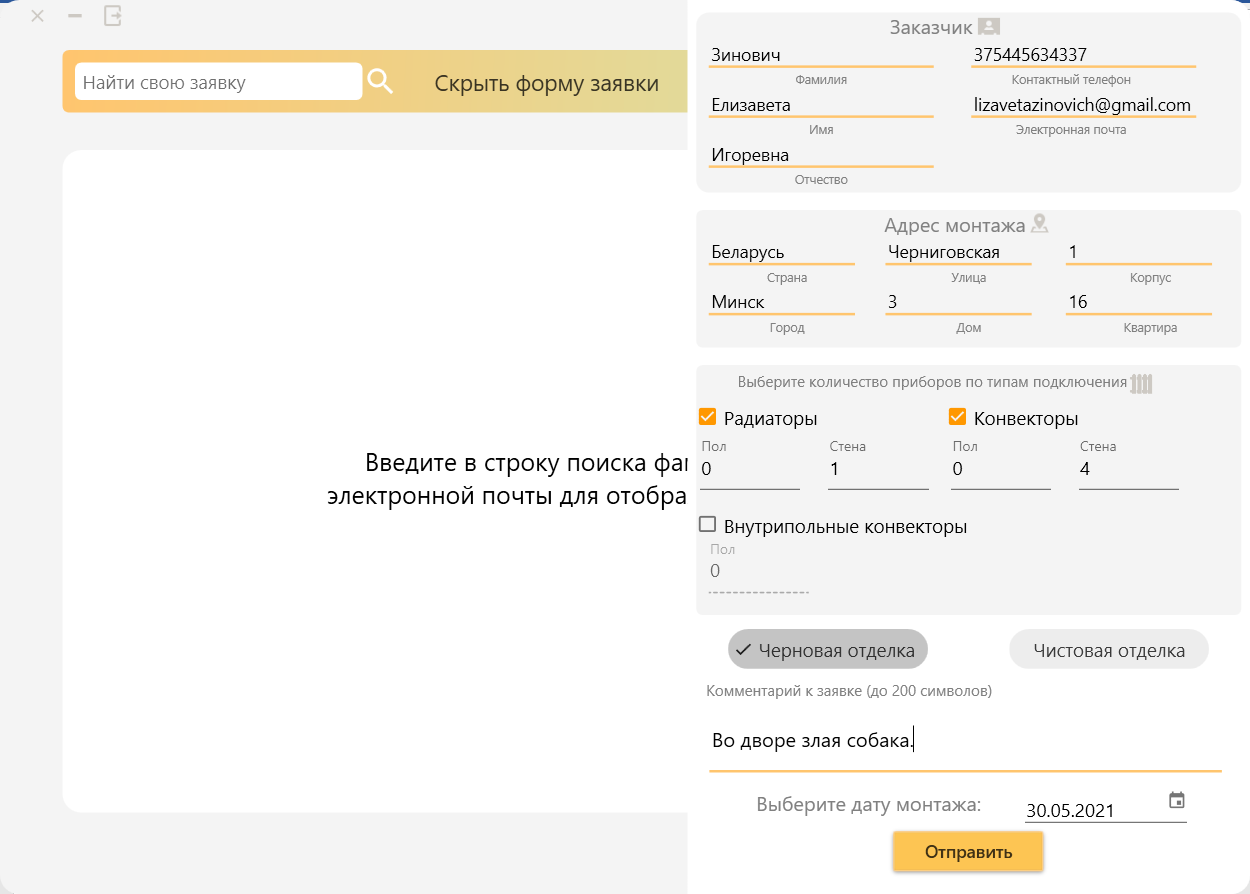
При прохождении валидации становится активной кнопка «Отправить», что изображено на рисунке 5.3.

Рисунок 5.3 Корректно заполненная форма заявки.

При нажатии на эту кнопку заявка клиента попадает в базу, а на указанный адрес электронной почты происходит попытка отправки письма.

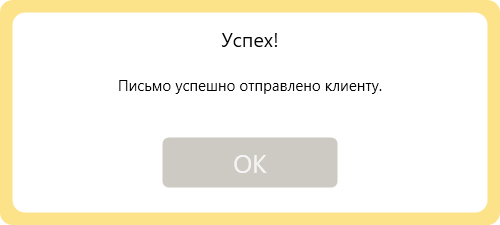
В случае успешной отправки письма выводится окно с оповещением, представленное на рисунке 5.4. Само содержание письма приведено в приложении Ж.

Рисунок 5.4 Сообщение об успешно отправленном письме.

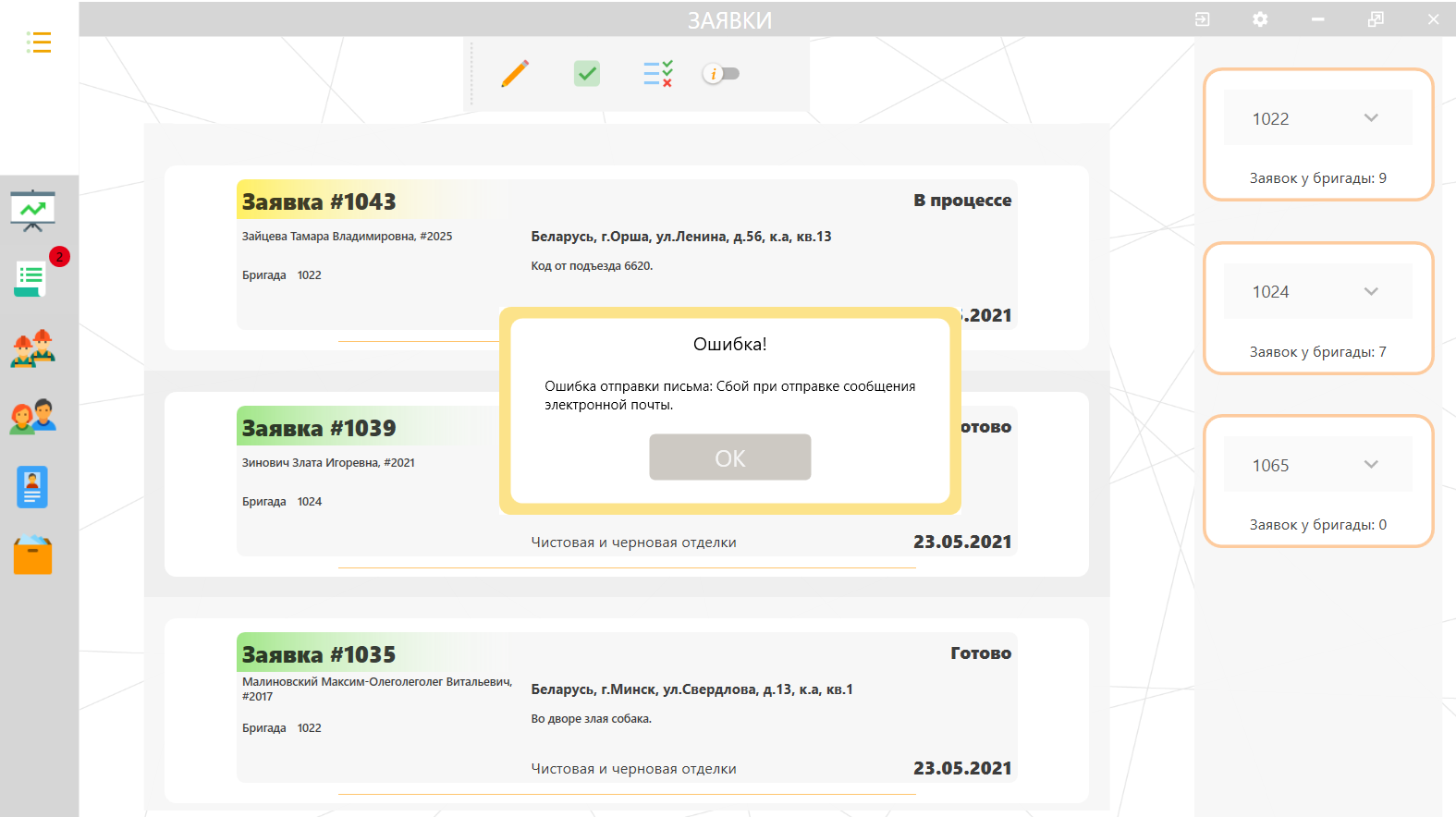
В случае, если письмо не было отправлено, выведется окно с оповещением, представленное на рисунке 5.5.

Рисунок 5.5 Сообщение об ошибке отправки письма.

# **Методика использования программного средства**

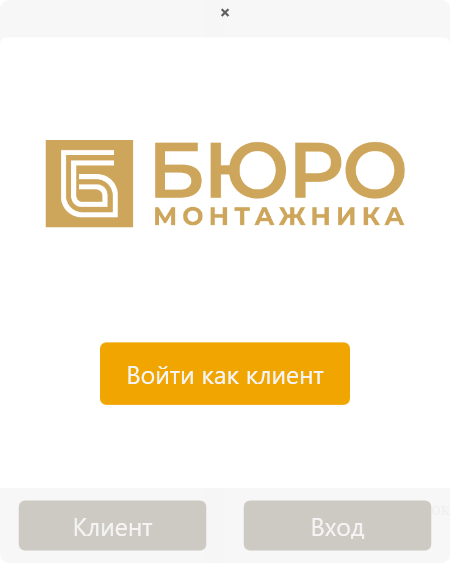
После запуска приложения в первую очередь открывается окно, в котором содержатся две вкладки: клиент и вход. Для входа под клиентом необходимо нажать на кнопку «Войти как клиент» (рисунок 6.1).

Рисунок 6.1 Вход как клиент (без авторизации).

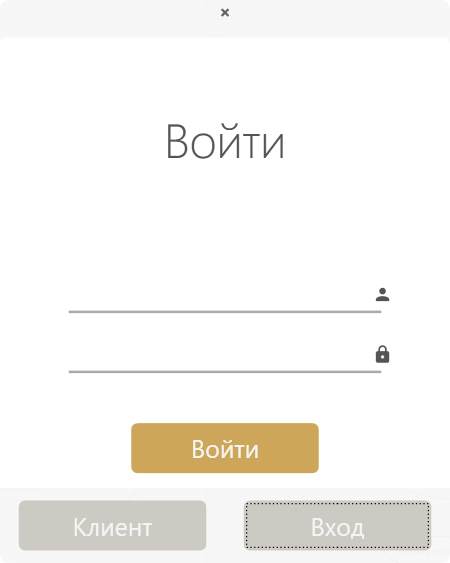
Если необходимо войти как бригада или администратор, необходимо нажать на кнопку «Вход» и перейти на вкладку со входом, представленную на рисунке 6.2.

Рисунок 6.2 Вход в аккаунт бригады или администратора.

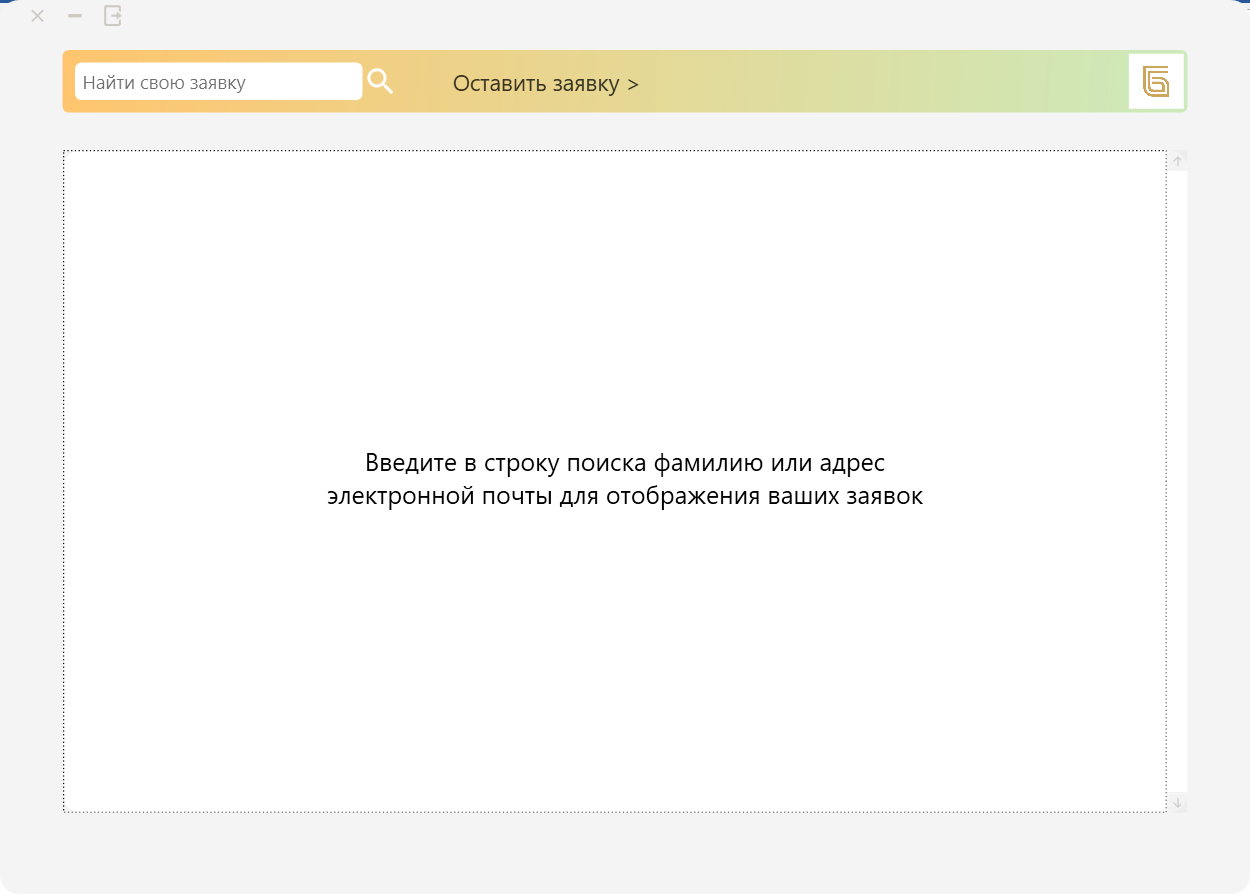
Окно клиента представлено на рисунке 6.3.

Рисунок 6.3 Окно клиента.

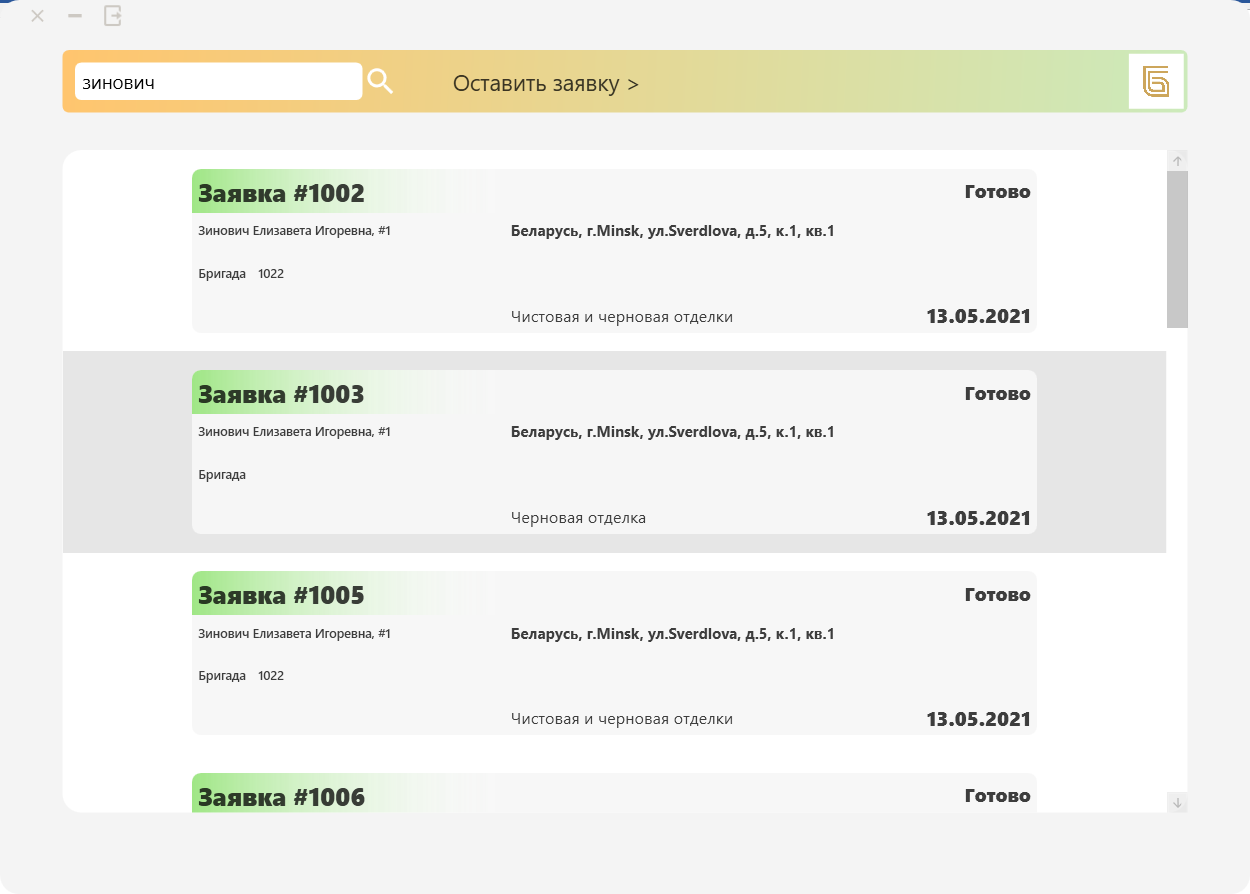
В строку поиска можно ввести данные клиента и получить информацию о его заявках (рисунок 6.4).

Рисунок 6.3 Поиск по фамилии клиента.

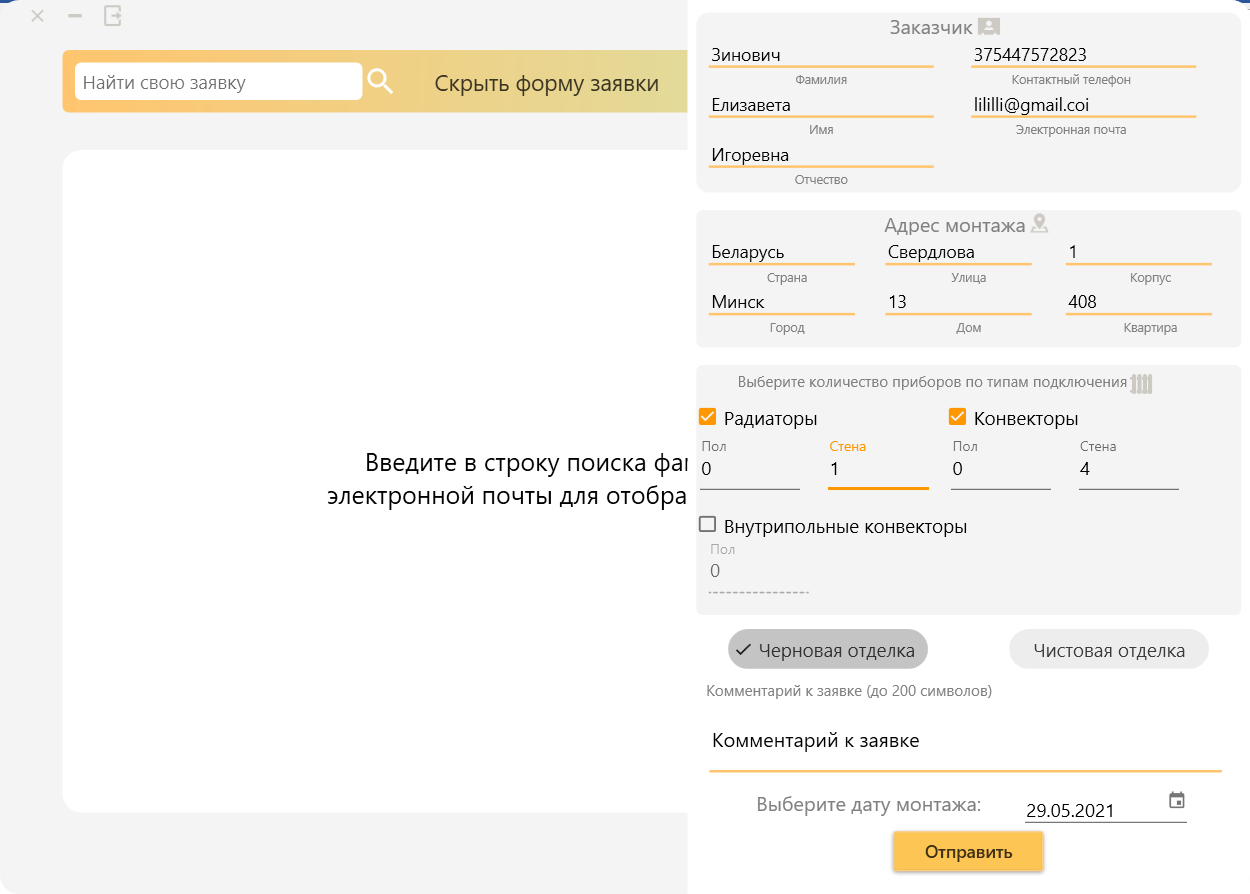
Если необходимо оставить заявку, нужно нажать на кнопку «Оставить заявку», после чего появится форма (рисунок 6.5).

Рисунок 6.5 Форма заявки клиента.

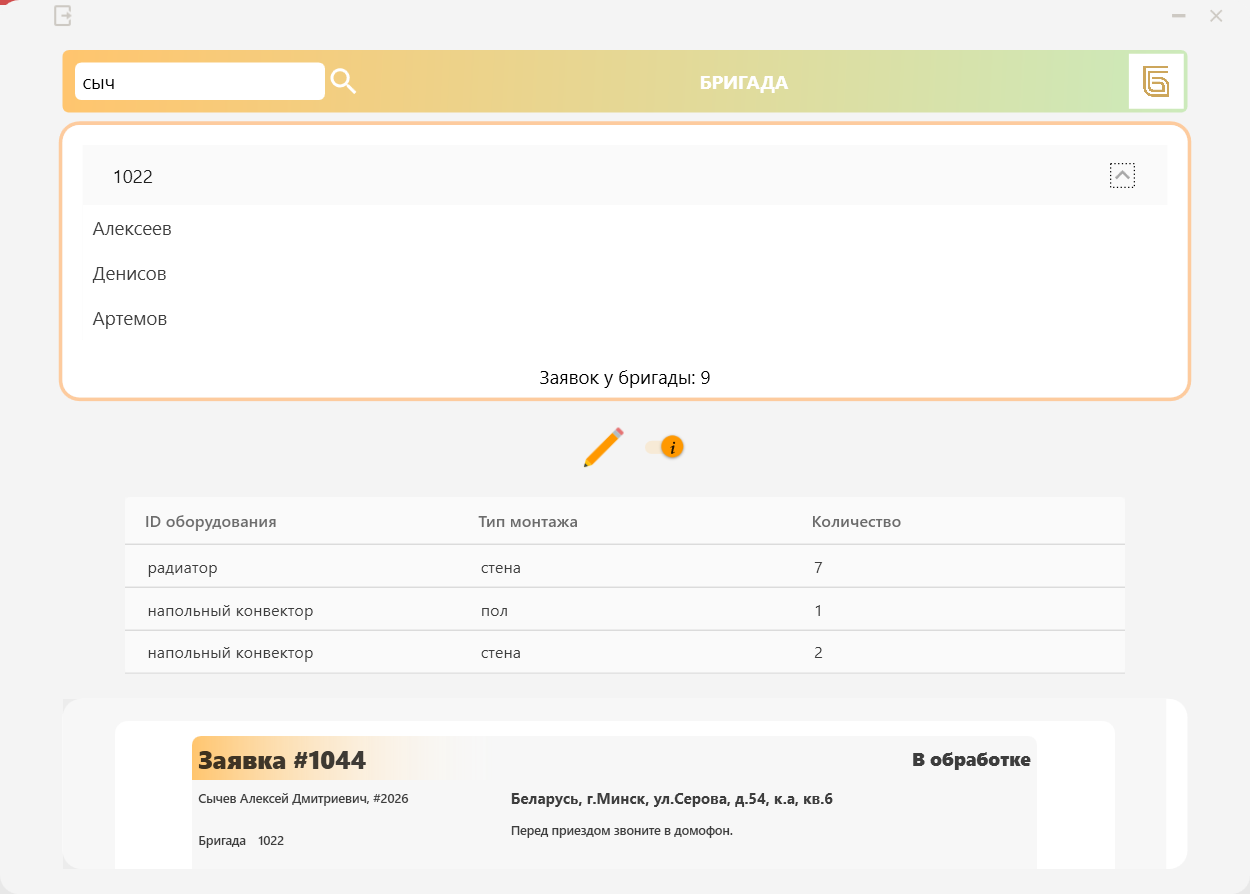
При входе в аккаунт бригады открывается главное окно бригады, представленное на рисунке 6.6. Здесь также можно осуществлять поиск, просматривать информацию о бригаде, об оборудовании конкретной заявки.

Рисунок 6.6 Окно бригады.

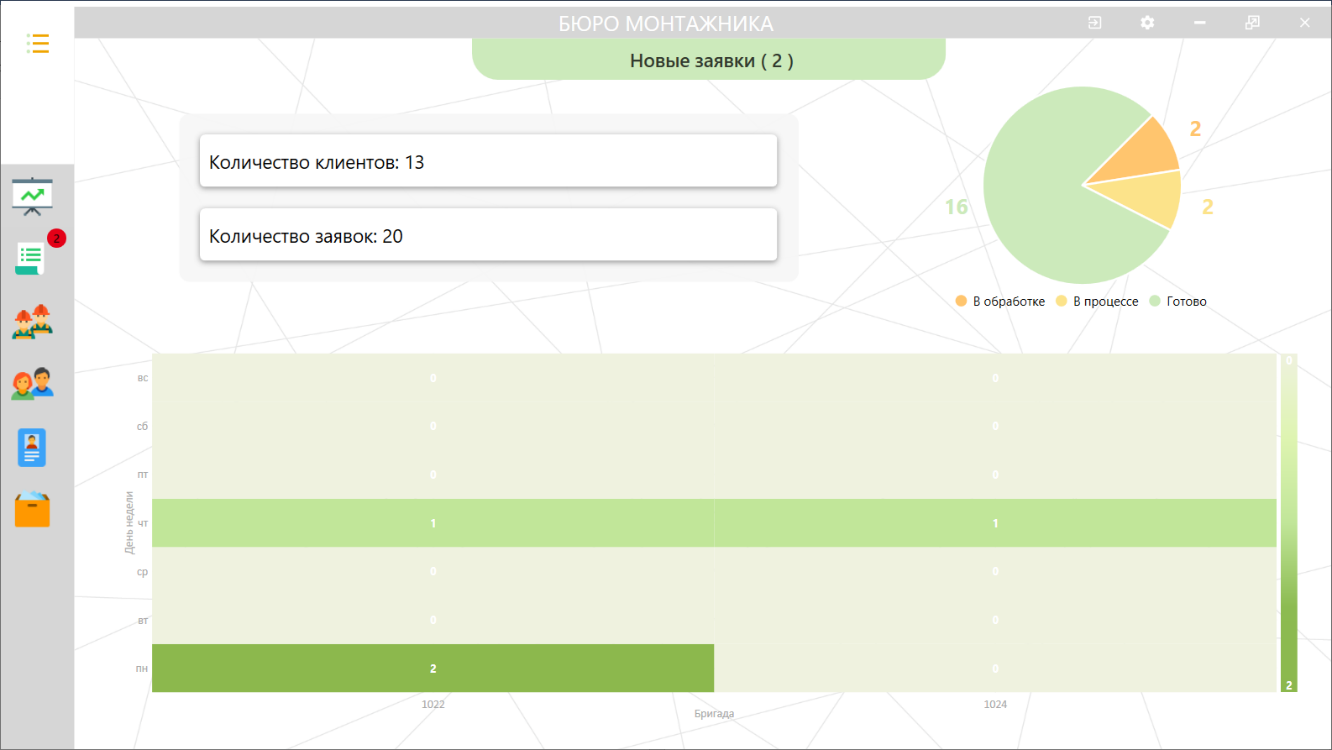
При входе в аккаунт администратора открывается главное окно администратора, представленное на рисунке 6.7 с активной вкладкой «Статистика».

Рисунок 6.7 Окно администратора, вкладка «Статистика».

Если перейти на вкладку «Заявки», изображенную на рисунке 6.8, можно совершить редактирование заявки (изменения её статуса, назначение бригаде) во всплывающем окне.

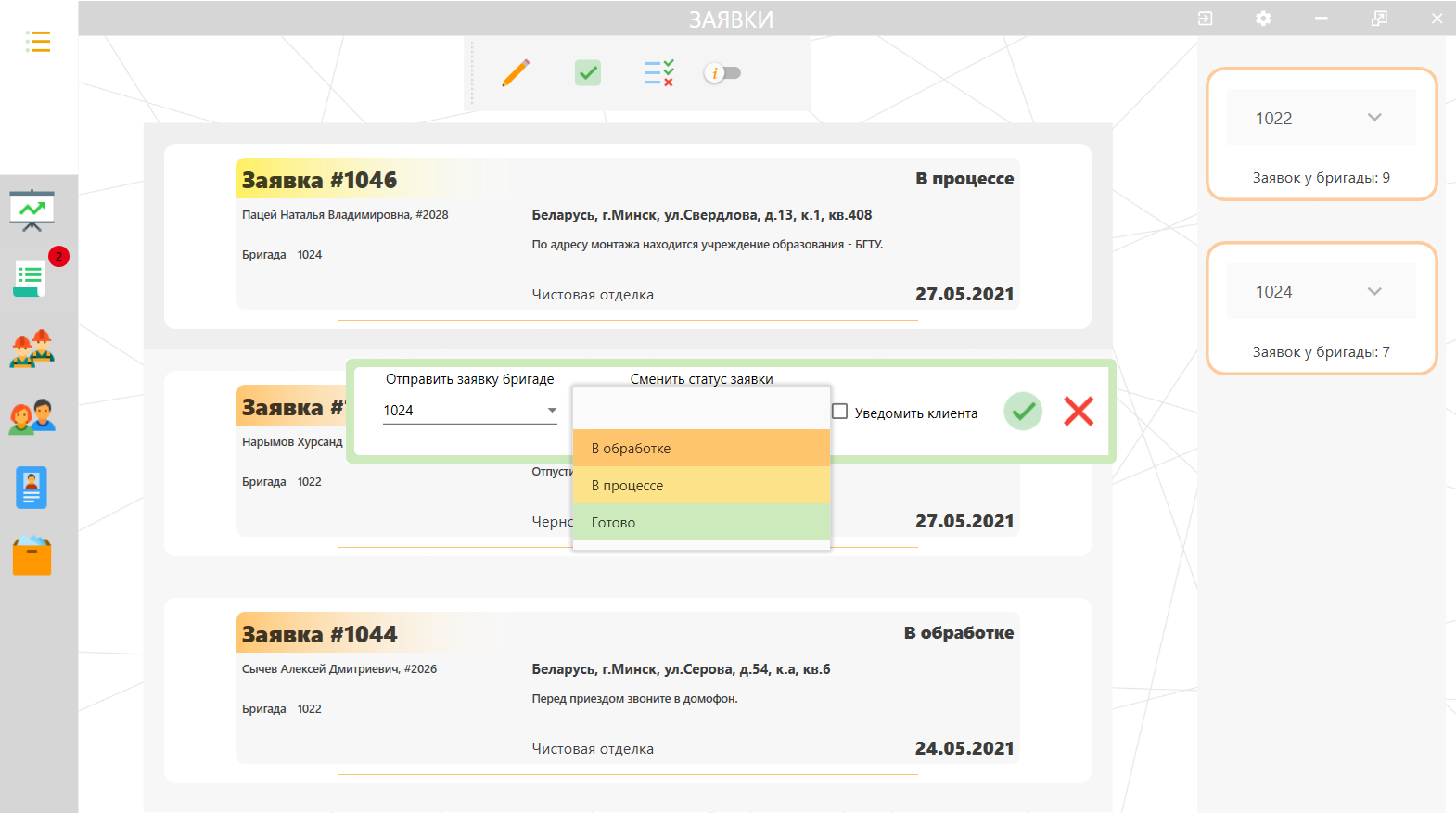
Изменения можно сохранить нажатием на зеленую кнопку с галкой или отменить нажатием на красный крест.

Рисунок 6.8 Окно администратора, редактирование заявки.

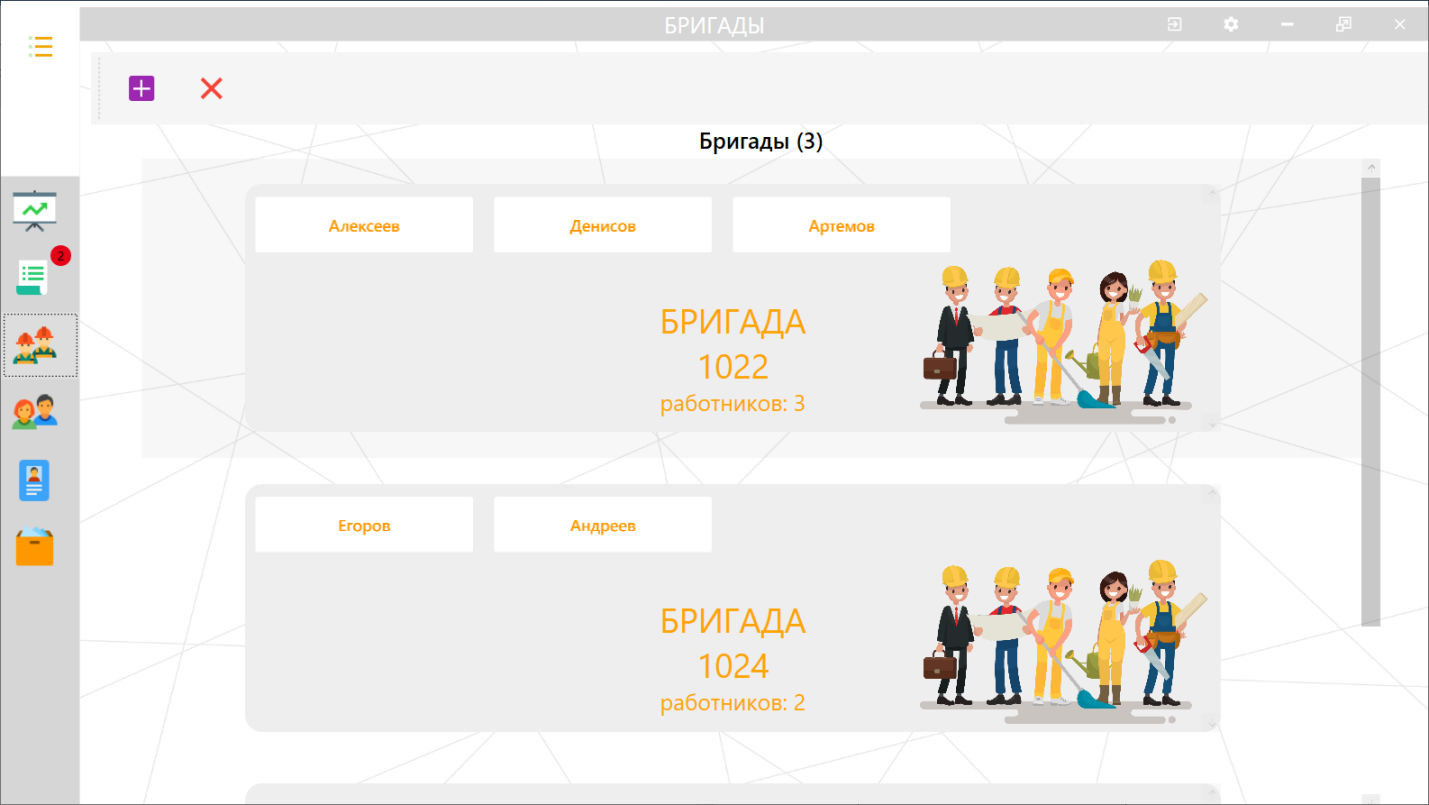
На вкладке «Бригады», изображенной на рисунке 6.9, можно добавить бригаду нажатием кнопки со знаком плюс или удалить выделенную бригаду нажатием на красный крест.

Рисунок 6.9 Окно администратора, вкладка «Бригады».

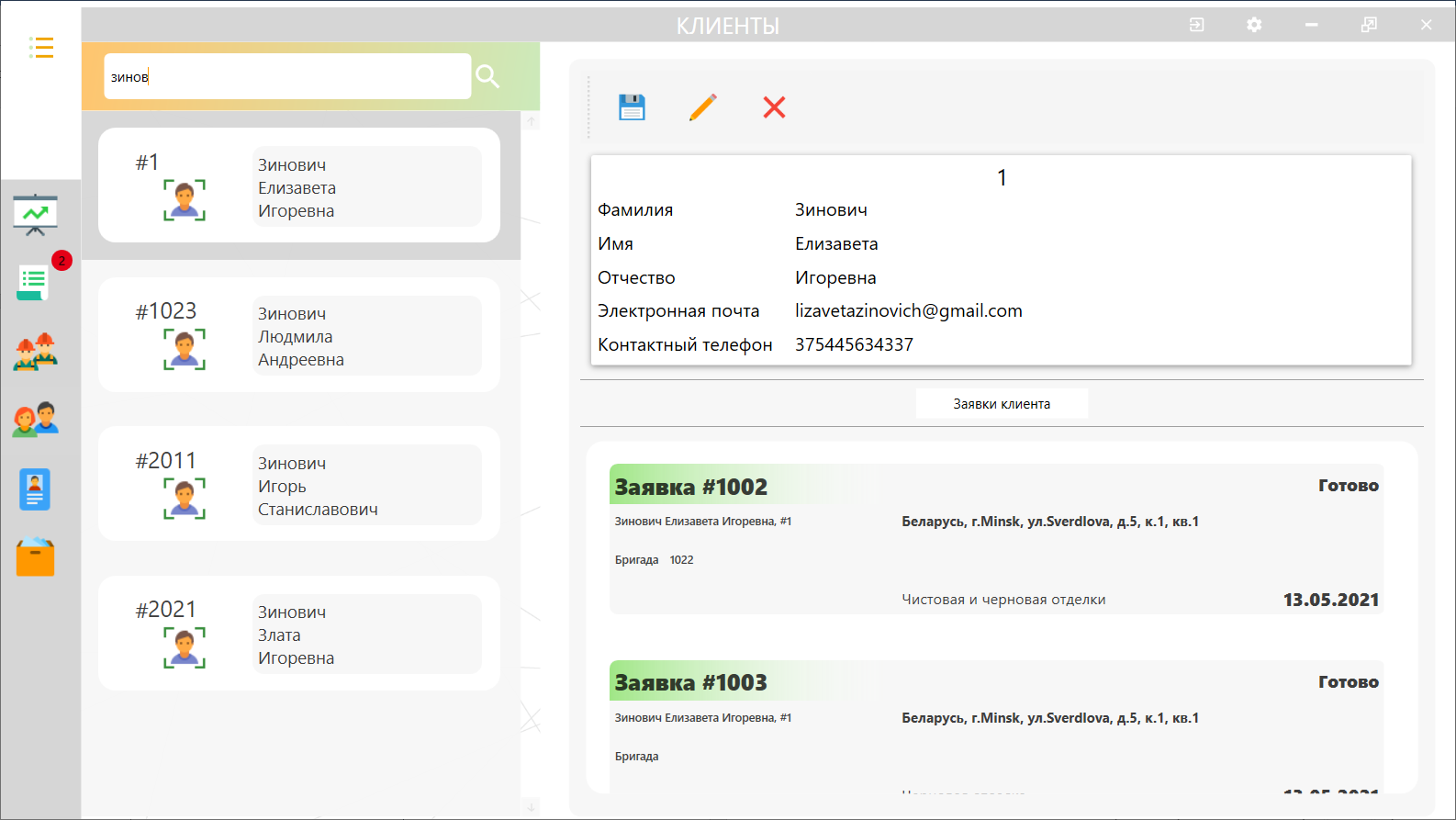
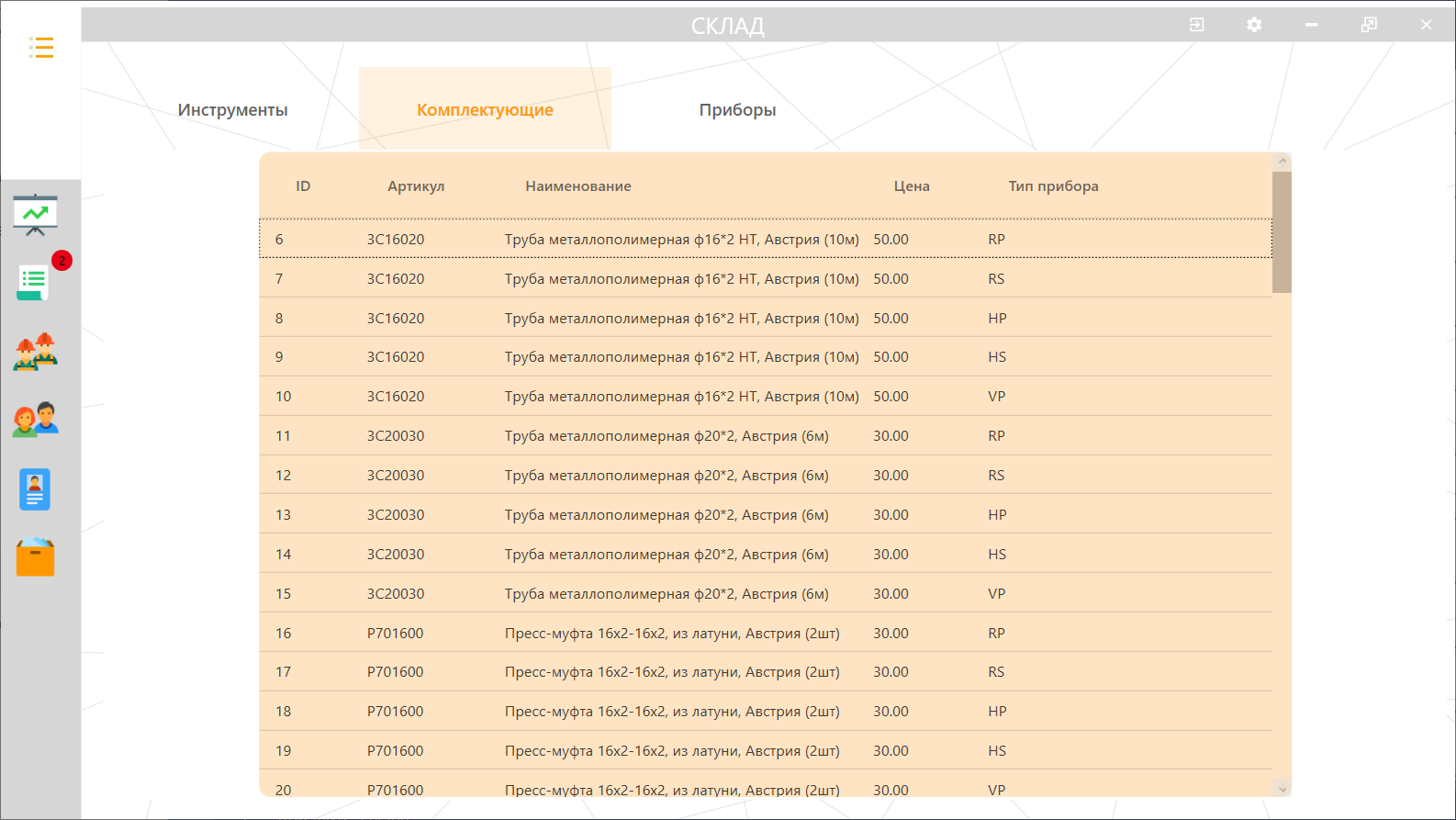
На вкладке «Клиенты», изображенной на рисунке 6.10, можно редактировать клиента нажатием на карандаш или удалить выделенного клиента.

Рисунок 6.10 Окно администратора, вкладка «Клиенты».

Вкладка «Работники» практически аналогична вкладке «Клиенты».

На вкладке «Склад», изображенной на рисунке 6.11, можно редактировать просмотреть список инструментов, комплектующих и приборов.

Рисунок 6.11 Окно администратора, вкладка «Склад».

# **ЗАКЛЮЧЕНИЕ**

По завершении выполнения курсовой работы было разработано программное средство «Организация монтажных работ – “Бюро Монтажника”».

Было изучено немало теоретического материала, а также просмотрено и разобрано большое количество уже готовых решений тех или иных задач.

В ходе выполнения курсовой работы было выполнено следующее:

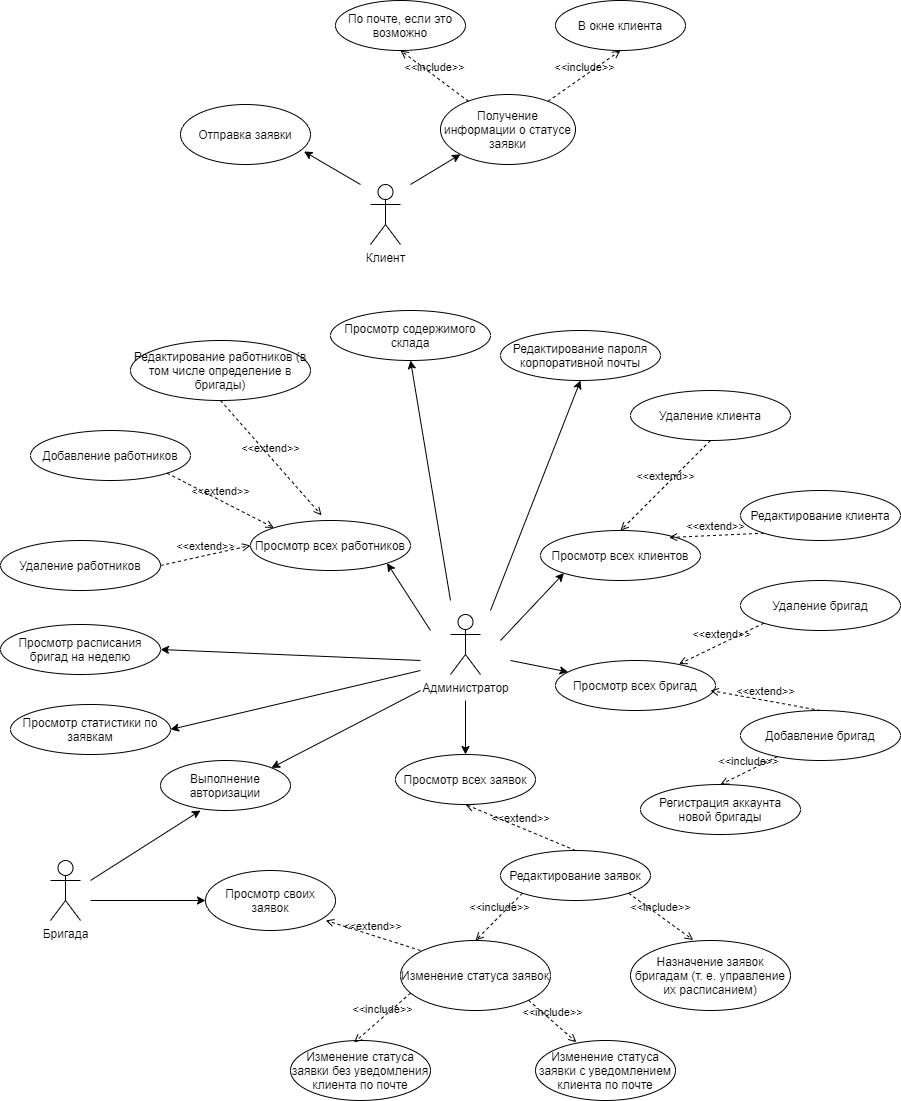
* разработка базы данных для хранения информации;
* разработка архитектуры приложения;
* разработка функциональных части приложения;
* разработка пользовательского интерфейса;
* написание исходного кода приложения;
* произведено тестирование приложения.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанное программное средство работает корректно, а требования технического задания выполнены в полном объеме.

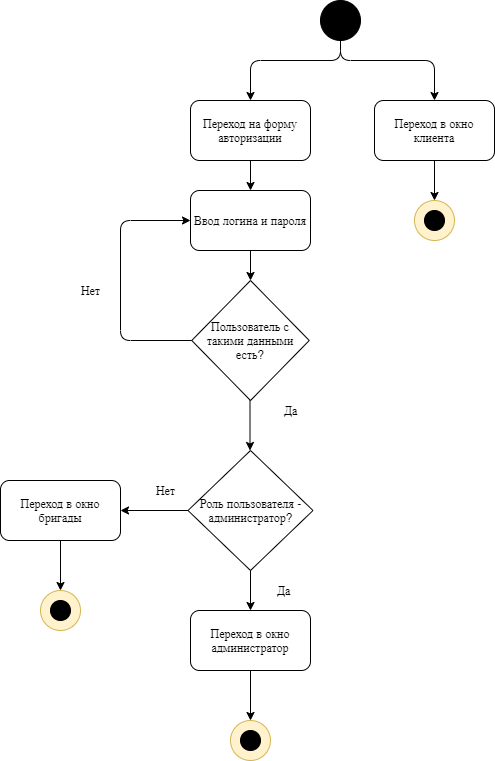
# **Список использованных источников**

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/>. Дата доступа: 20.04.2021
4. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com>. Дата доступа: 20.05.2021
5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2021

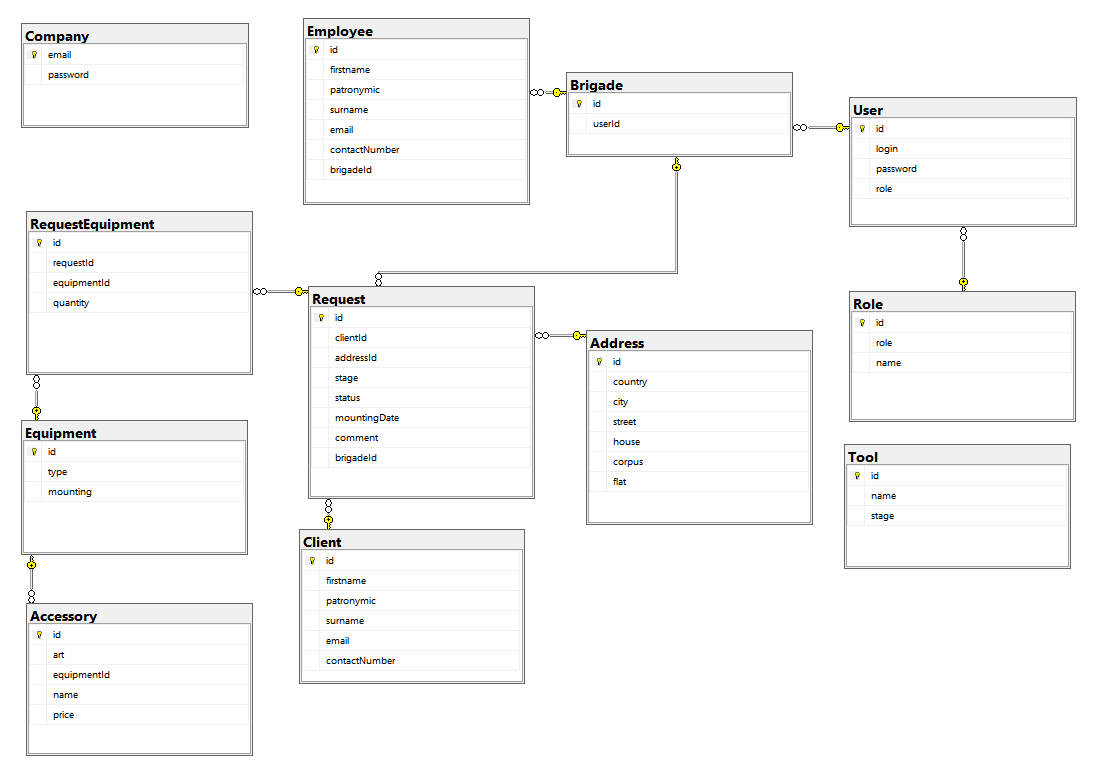
# **Приложение А**

Рисунок 1 Диаграмма вариантов использования

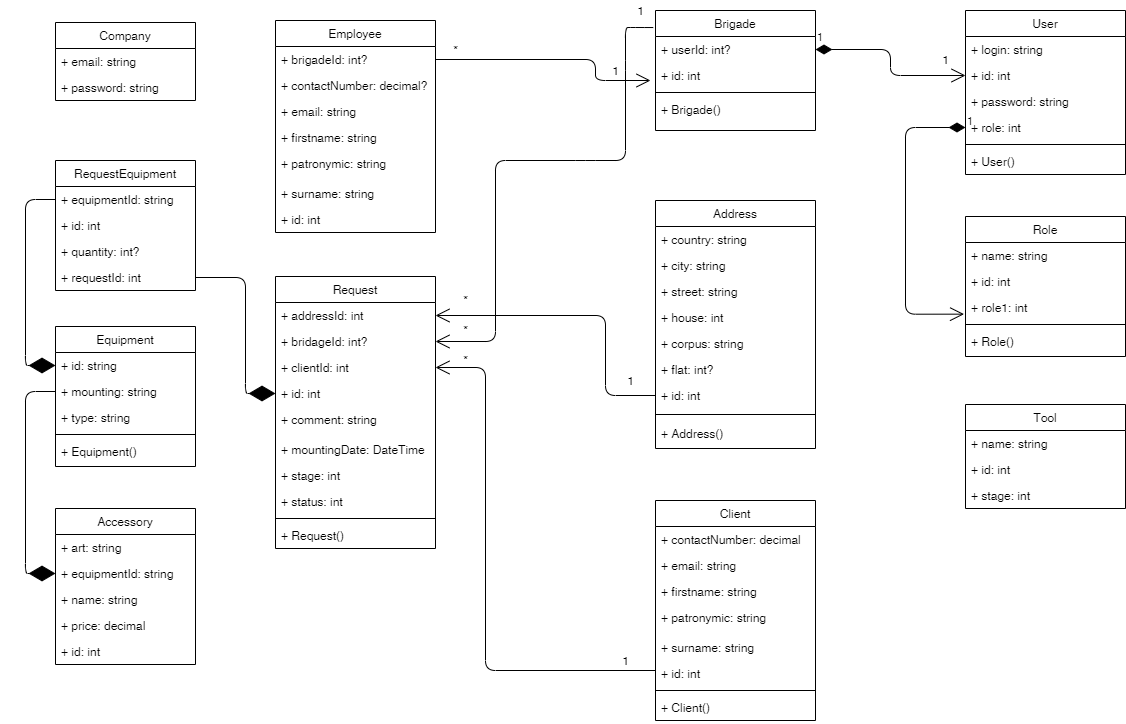
# **Приложение Б**

Рисунок 2 Диаграмма деятельности, описывающая процесс входа в приложение под разными ролями

# **Приложение В**

Рисунок 3 Логическая схема базы данных

# **Приложение Г**

Рисунок 4 Диаграмма классов

# **Приложение Д**

Рисунок 5 Реализация класса RelayCommand

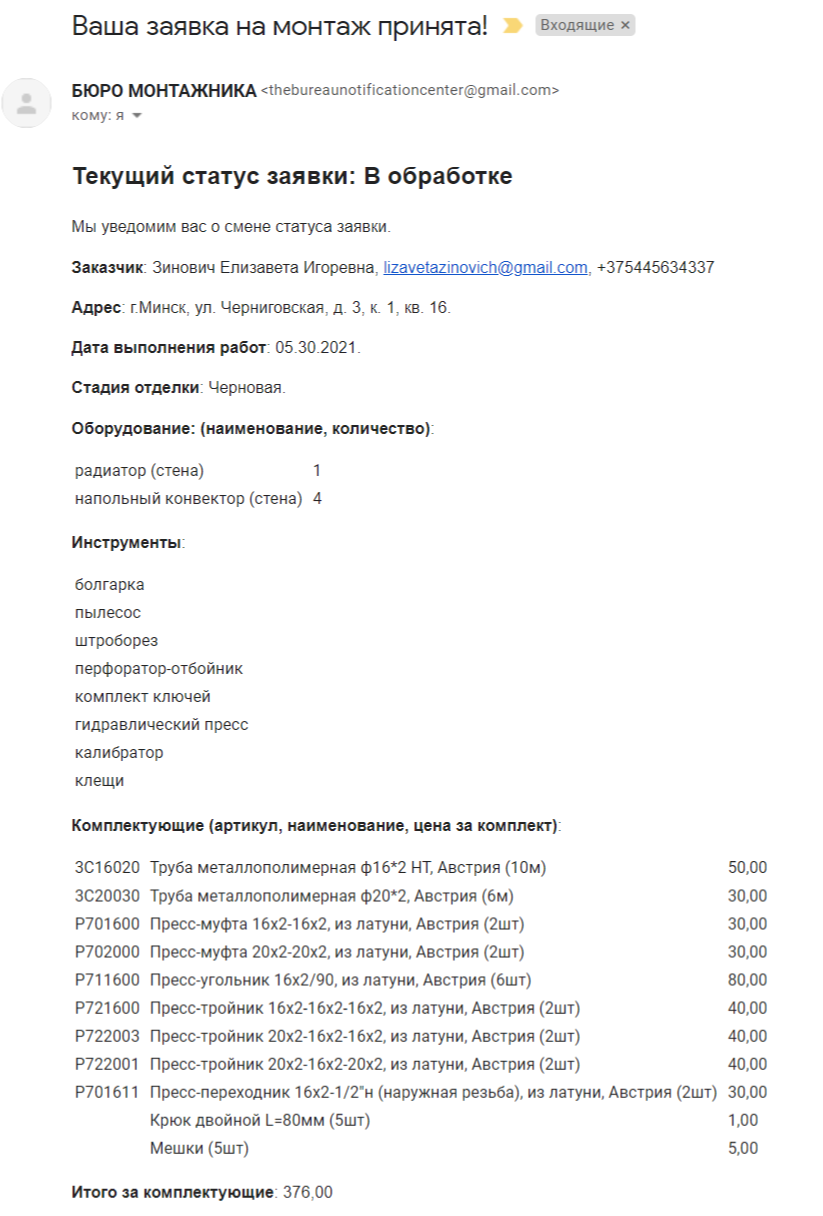


# **Приложение Е**

Рисунок 6 Реализация класса ViewModelBase



# **Приложение Ж**

Рисунок 7 Письмо с подтверждением заявки