МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 05 01-03 Информационные системы и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство «Автосервис «Repair Auto Services»»

Исполнитель

студент (ка) 2 курса группы 1 Шумова Елизавета Игоревна

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2022

**Содержание**

[Введение 3](#_Toc104858117)

[1. Аналитический обзор литературы и постановка задачи 4](#_Toc104858118)

[1.1 Анализ прототипов 4](#_Toc104858119)

[1.2 Постановка задачи 6](#_Toc104858120)

[2. Анализ требований к программному средству и разработка функциональных требований 7](#_Toc104858121)

[2.1 Определение требований к программному средству 7](#_Toc104858122)

[2.2 Описание средств разработки 7](#_Toc104858123)

[2.3 Описание функциональности программного средства 8](#_Toc104858127)

[2.4 Спецификация функциональных требований 8](#_Toc104858132)

[3. Проектирование и создание программного средства 10](#_Toc104858135)

[3.1 Архитектура системы 10](#_Toc104858139)

[3.2 Диаграммы UML 10](#_Toc104858140)

[3.3 Проектирование логической структуры базы данных 11](#_Toc104858141)

[3.4 Структура проекта 12](#_Toc104858142)

[4. Реализация программного средства 14](#_Toc104858146)

[4.1 Реализация MVVM 14](#_Toc104858147)

[4.2 Реализация регистрации и авторизации пользователей 16](#_Toc104858148)

[4.3 Создание и редактирование заказа 20](#_Toc104858154)

[4.4 Создание отзывов 22](#_Toc104858161)

[5. Тестирование и проверка работоспособности 23](#_Toc104858166)

[5.1 Тестирование регистрации и авторизации 23](#_Toc104858167)

[5.2 Тестирование добавления объектов базы данных 25](#_Toc104858168)

[6. Руководство по использованию 26](#_Toc104858174)

[Заключение 34](#_Toc104858175)

[Список литературы 35](#_Toc104858176)

[Приложение А 36](#_Toc104858177)

[Приложение Б 38](#_Toc104858178)

[Приложение В 39](#_Toc104858179)

[Приложение Г 40](#_Toc104858180)

# **Введение**

Рынок автомобилей с каждым годом растет, количество автомобилей увеличивается, однако каждый автомобиль имеет высокий шанс получить повреждения. Для устранения неисправностей, замены деталей и тюнинга автовладельцы обращаются к опытным специалистам автосервисов.

В наши дни люди ценят свое время, а программные средства позволяют его экономить. Поэтому темой курсового проекта является разработка программного средства для автосервиса «Repair Auto Services». Оно должно позволить пользователям в лице клиентов оформлять заказы по ремонту автомобиля, отслеживать статус заказа. Пользователи в лице администраторов могут изменять статус заказов и уведомлять пользователей-клиентов об окончании заказанной работы.

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами программных средств выбранной мной темы;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

# **Аналитический обзор литературы и постановка задачи**

Для того чтобы окончательно определиться с постановкой задачи моего курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

## **1.1 Анализ прототипов**

АСУ-Автосервис – программное средство для автоматизации СТО, разработанная командой Toolhand (рисунок 1.1).

Достоинства:

* удобный и понятный интерфейс;
* три варианта версий (Lite, Business и Pro);
* возможность СМС-рассылок.

Недостатки:

* в бесплатной Lite версии однопользовательский режим и ограниченное количество возможных записей в базу данных.

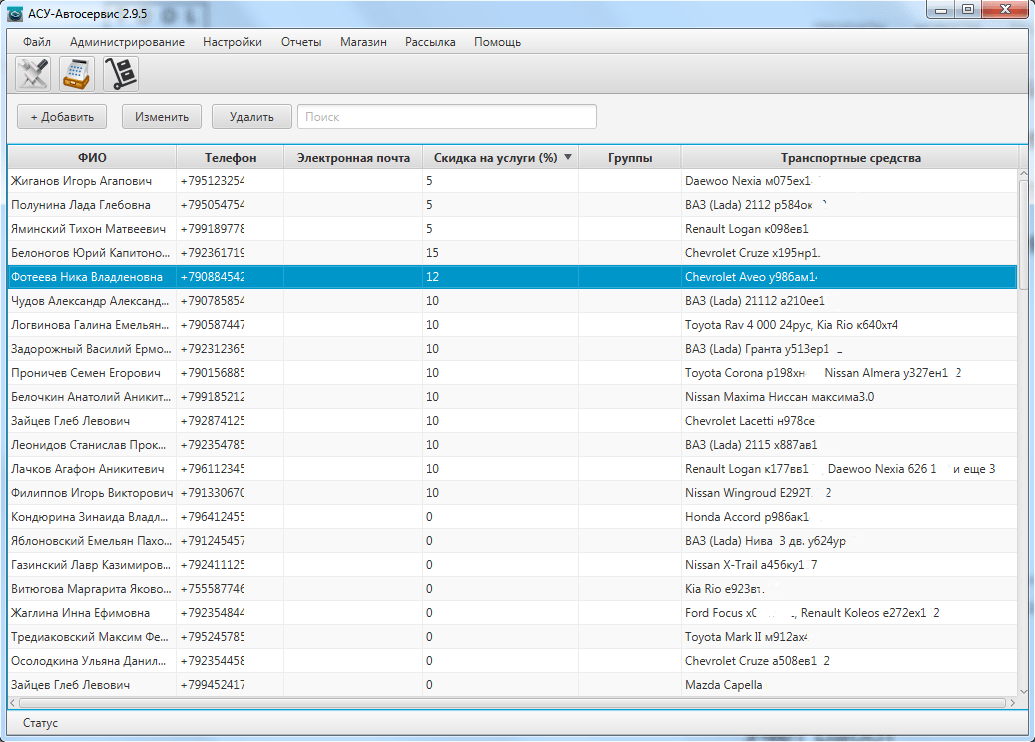


Рисунок 1.1 – Интерфейс ПС «АСУ-Автосервис»

Корс-Гараж – программное средство для автоматизации автосервисов, разработанная Kors (рисунок 1.2).

Достоинства:

* простой интерфейс;
* импорт из Excel;
* настройка меню печать.

Недостатки:

* неэстетичный дизайн;
* функциональные возможности редко дорабатываются и обновляются разработчиками.

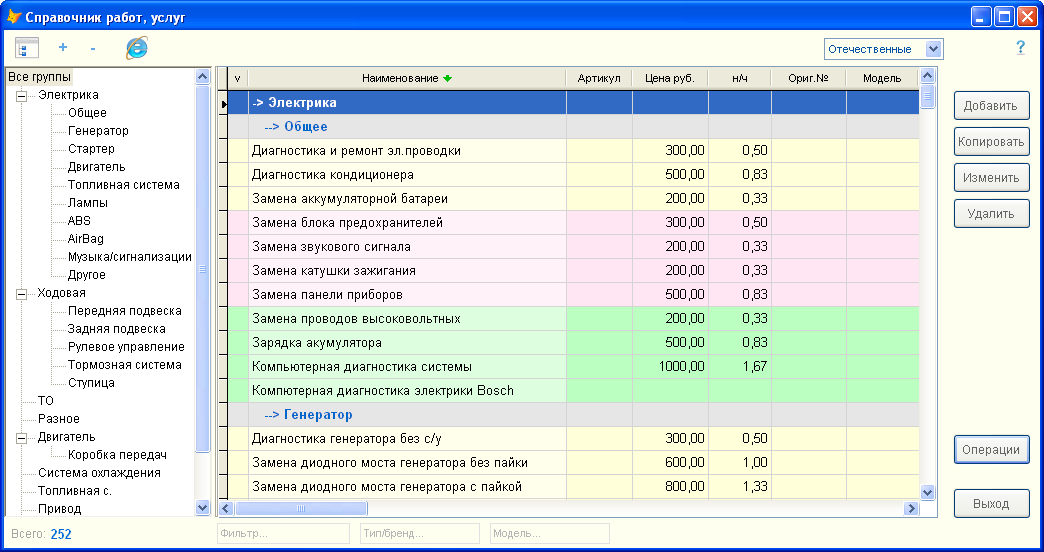


Рисунок 1.2. – Интерфейс ПС «Корс-Гараж»

STOCRM – учетная программа для автосервиса (рисунок 1.3).

Достоинства:

* современный и практичный дизайн;
* анализ услуг и заказов;
* возможность настройки графика работы сотрудников;
* встроенная телефония.

Недостатки не выявлены.

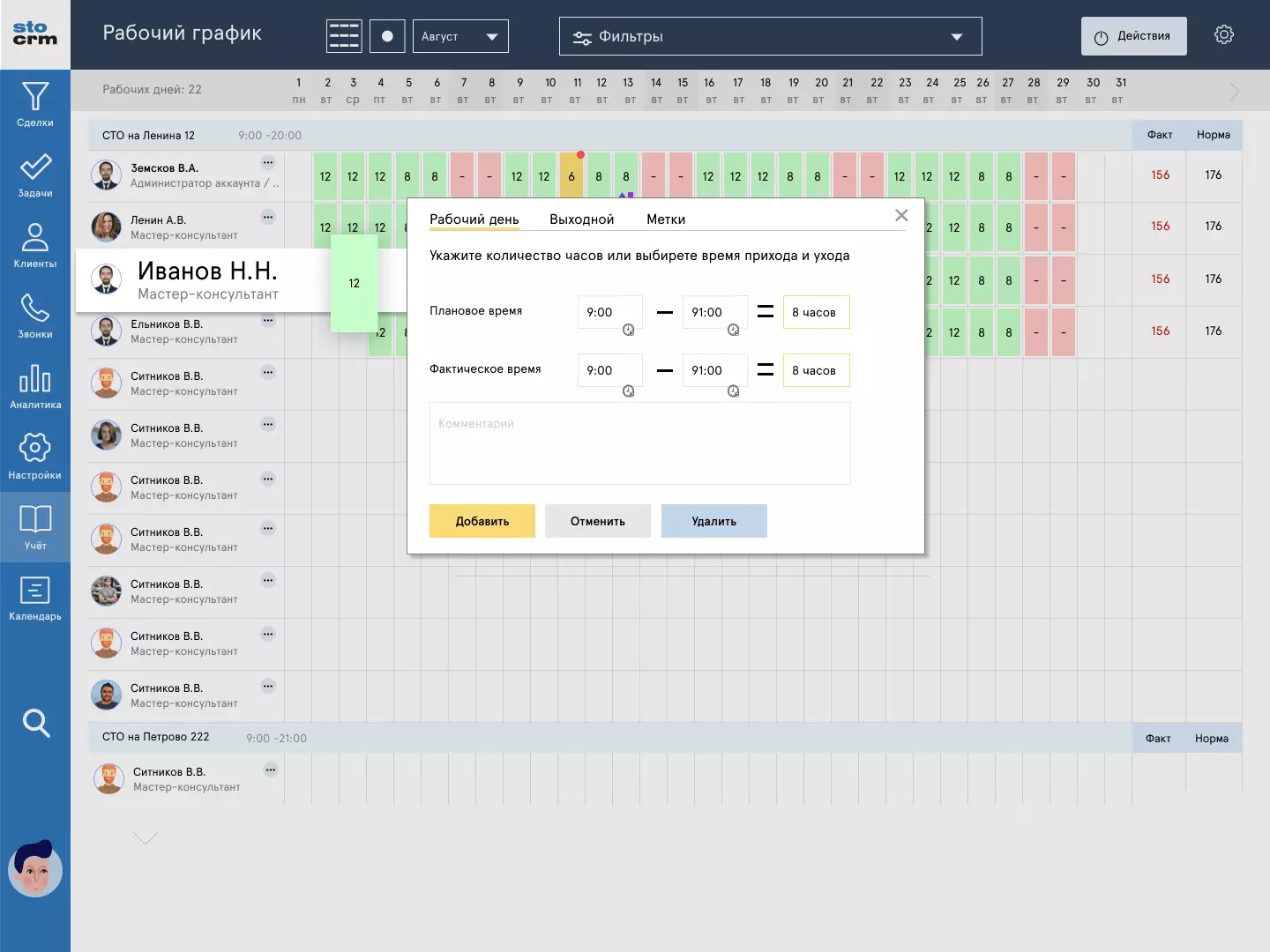


Рисунок 1.3. – Интерфейс ПС «STOCRM»

## **1.2 Постановка задачи**

В ходе анализа прототипов, были выделены основная задача и функциональные требования проекта.

Основной задачей курсового проекта является разработка десктопного приложения, позволяющего выполнять авторизацию и регистрацию пользователей; пользователям в лице администратора добавлять, удалять и редактировать заказы, уведомлять пользователей-клиентов о состоянии заказа. Пользователю в лице клиента просматривать информацию о мастерах, оставлять отзывы и заполнять форму заказа.

Функциональные требования описаны в главе 2.

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

Для того, чтобы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

## **2.1 Определение требований к программному средству**

Функционально ПС должно выполнять следующие задачи:

* регистрация и авторизация пользователей;
* сохранение рабочей информации в базе данных;
* для администратора добавление сотрудников и услуг;
* для пользователя оформление заказов;
* для пользователя возможность оставлять отзывы;
* поиск по заказам.

## **Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2019;
* программная платформа .NET Framework 4.8;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework 6.0.0;
* Microsoft SQL Server 2019;

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология Entity Framework. Она предоставляет собой объектно-ориентированную технологию доступа к данным, является object-relational mapping (ORM) решением для платформы .NET Framework. Entity Framework предоставляет возможность взаимодействия с объектами посредством LINQ to Entities. Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.



## **Описание функциональности программного средства**

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования (Приложение А).

Спецификация функциональности программного средства представлена в следующем пункте.



## **Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В программном средстве при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются адрес электронной почты и пароль пользователя, которые содержатся в базе данных. Для того, чтобы зарегистрироваться необходимо ввести пароль, адрес электронной почты, имя и фамилию. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных.

При авторизации администратора на странице «*Сотрудники*» он может добавлять и удалять сотрудников. На странице «*Услуги*» он может добавлять и удалять услуги. На странице «*Пользователи*» он может просматривать список пользователей. На странице «*Заказы*» администратор может просматривать список заказов, редактировать их, осуществлять поиск по номеру заказа, заказавшему пользователю и фильтровать заказы по статусу. Также на этой странице администратор может оформлять заказ. На странице «*Автомобили*» он может добавлять и удалять производителей и модели авто.

При авторизации пользователя он может на странице «*Главная*» изменять личную информацию, добавлять автомобили, а также просматривать уведомления о статусе заказа. На странице «*Заказы*» он может просматривать список заказов, редактировать их (если они приняты, но еще не выполняются), осуществлять поиск по номеру заказа, фильтровать заказы по статусу и осуществлять новые заказы. На странице «*Отзывы*» пользователь может оставлять собственные отзывы и просматривать отзывы других пользователей. На странице «*Сотрудники*» он может просматривать список сотрудников автосервиса.

Кнопка для выхода их приложения должна возвращать пользователя к окну авторизации.

Все данные о сотрудниках, услугах, пользователях и заказах должны заноситься в базу данных в соответствующие таблицы.



# **Проектирование и создание программного средства**

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



## **3.1 Архитектура системы**

В данном приложении используется архитектурный паттерн MVVM (MODEL-VIEW-VIEWMODEL). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

* Модель описывает используемые в приложении данные.
* Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.
* Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

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



Рисунок 3.1 – Структура шаблона MVVM

## **3.2 Диаграммы UML**

UML — унифицированный язык моделирования (Unified Modeling Language) — это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Его можно использовать для визуализации, спецификации, конструирова­ния и документирования программных систем.

Диаграмма — это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования.

Диаграмма последовательности взаимодействия с окном авторизации приведена в приложении Б.

Диаграммы использования показывает актеров и возможные прецеденты, при работе с приложением. Диаграмма использования приведена в приложении А.

Диаграмма классов служит для представления статической структуры мо­дели системы в терминологии классов объектно-ориентированного программирования.

Диаграмма классов ViewModel приведена в приложении В.

## **3.3 Проектирование логической структуры базы данных**

Для реализации поставленной задачи была создана база данных. Для её создания использовались система управления реляционными базами данных Microsoft SQL Server 2019.

База данных – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами. База данных нашей программной подсистемы состоит из 14 таблиц, ее схема изображена в приложении Г.

Таблица Admins хранит логин и пароль администратора и содержит два столбца: Login и Password.

Таблица Customers хранит информацию о заказчиках и содержит 6 столбцов: Id – идентификатор пользователя, Name – его имя, Surname – фамилия, Email – адрес электронной почты, Password – пароль, Image – фотография профиля.

Таблица Notifications хранит информацию для уведомлений о состоянии заказа и содержит 4 столбца: Id – идентификатор, Message – сообщение уведомления, CreatedAt – дата, CustomerId – идентификатор пользователя.

Таблица Reviews хранит отзывы пользователей и содержит 5 столбцов: Id – идентификатор, Title – заголовок отзыва, Description – текст отзыва, CreatedAt – дата создания отзыва, CustomerId – идентификатор пользователя.

Таблица Cars хранит информацию об автомобилях и содержит 6 столбцов: Id – идентификатор, CreatedYear – год выпуска автомобиля, Mileage – пробег автомобиля, Image – фотография, CustomerId – идентификатор пользователя, ModelId – идентификатор модели автомобиля.

Таблица ModelTypes хранит информацию о типах моделей автомобилей и содержит два столбца: Id – идентификатор, Name – название модели.

Таблица Producers хранит информацию о производителях автомбилей и содержит три столбца: Id – идентификатор, Name – название компании, Country – страна производителя.

Таблица Models хранит информацию о моделях автомобилей и содержит 6 столбцов: Id – идентификатор, Name – название модели, Power – мощность, AccelerationSec ModelTypeId – идентификатор типа модели, ProducerId – идентификатор производителя.

Таблица Orders хранит информацию о заказах и содержит 7 столбцов: Id – номер заказа, CreatedAt – дата создания заказа, StartDate – дата начала заказа, Sum – сумма заказа, CarId – идентификатор автомобиля, EmployeeId – идентификатор сотрудника, OrderStatusId – идентификатор статуса заказа.

Таблица ServiceOrders используется для связи услуг и заказов. Содержит два столбца: Service\_id – идентификатор услуги и Order\_id – номер заказа.

Таблица Services хранит информацию об услугах и содержит три столбца: Id – идентификатор услуги, Name – наименование услуги, Price – цена услуги.

Таблица OrderStatus хранит информацию о статусе заказа и содержит два столбца: Id – идентификатор, Name – статус заказа.

Таблица Employees хранит информацию о сотрудниках и содержит 4 столбца: Id – идентификатор сотрудника, Name – имя сотрудника, Surname – фамилия, Experience – стаж работы.

Таблица MigrationHistory хранит информацию о миграциях. Миграции – механизм работы с базами данных, позволяющий вносить изменения в схему базы данных без потерь данных. Таблица MigrationHistory содержит 4 столбца: MigrationId – идентификатор миграции, ContextKey – ключ контекста, Model – модель, ProductVersion – версия продукта.

## **3.4 Структура проекта**

Структура проекта представлена на рисунке 3.2.

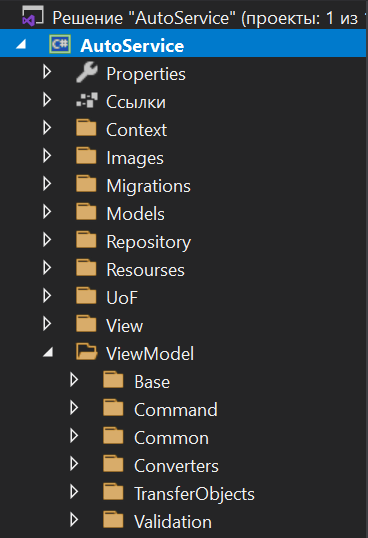


Рисунок 3.2 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| App.config | Файл с параметрами проекта. |
| Папка Context | Папка, содержащая контекст базы данных. |
| Папка Images | Папка, содержащая все изображения. |
| Папка Migrations | Папка, содержащая миграции базы данных. |
| Папка Models | Папка, содержащая классы моделей. |
| Папка Repository | Папка, содержащая паттерн репозиторий. |
| Папка Resources | Папка, содержащая словарь ресурсов. |
| Папка UoF | Папка, содержащая паттерн Unit of Work |
| Папка View | Папка, содержащая классы представлений. |
| Папка ViewModel | Папка, содержащая классы моделей представлений. |
| Папка Base | Папка, содержащая базовый класс ViewModel – BaseViewModel. |
| Папка Command | Папка, содержащая паттерн Command. |
| Папка Common | Папка, содержащая класс загрузки изображений LoadImage, класс отображения сообщений ShowMessage, класс текущего пользователя CurrentUser, интерфейсы для загрузки изображений и отображения сообщений. |
| Папка Converters | Папка, содержащая классы преобразований типов данных BooleanConverter, BoolToVisibilityConverter, DateConverter. |
| Папка TransferObjects | Папка, содержащая паттерн Transfer Objects. |
| Папка Validation | Папка, содержащая классы для валидации данных. |
| Папка Enums | Папка, содержащая перечисления. |



# **Реализация программного средства**

## **4.1 Реализация MVVM**

Паттерн MVVM реализуется через базовый класс BaseViewModel, который реализует интерфейс INotifyPropertyChanged и от которого наследуются все страницы. Реализация класса BaseViewModel приведена в листинге 4.1.

namespace AutoService.ViewModel.Base

{

public abstract class BaseViewModel : INotifyPropertyChanged

{

public Action Close { get; set; }

public RelayCommand CloseCommand

{

get

{

return new RelayCommand((o) => Close());

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

}

}

}

Листинг 4.1 – Реализация класса BaseViewModel

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта (листинг 4.2). В WPF команды представлены интерфейсом ICommand. Класс реализует два метода:

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

Листинг 4.2 – Реализация класса RelayCommand

namespace AutoService.ViewModel.Command

{

public class RelayCommand : ICommand

{

private Action<object> \_execute;

private Func<object, bool> \_canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null || \_canExecute(parameter);

}

public void Execute(object parameter)

{

\_execute(parameter);

}

}

}

## **Реализация регистрации и авторизации пользователей**

Для того чтобы пользоваться приложением каждый пользователь должен войти в систему, предварительно зарегистрировавшись.

В листинге 4.3 приведен код, реализующий регистрацию. Для безопас­ности хранения информации используется хэширование.

Листинг 4.3 – Реализация регистрации

public RelayCommand RegisterCommand

{

get

{

return \_registerCommand ?? (\_registerCommand = new RelayCommand((o) =>

{

var box = o as PasswordBox;

Register.Password = box.Password;

if(!Register.HasErrors)

{

if (\_unitOfWork.CustomerRepository.GetEntityByConditionOrDefault(c => c.Email == Register.Email) is null)

{

var customer = new Customer

{

Id = Register.Id,

Name = Register.Name,

Surname = Register.Surname,

Email = Register.Email,

Password = BCrypt.Net.BCrypt.HashPassword(Register.Password),

Cars = new List<Car>(CustomerCars),

Image = Register.Image

};

\_unitOfWork.CustomerRepository.Add(customer);

\_unitOfWork.SaveChanges();

CurrentUser.Id = customer.Id;

var main = new MainWindow();

main.Show();

this.Close();

}

else

{

\_showMessage.Show(

"Введенный Email уже используется",

"AutoService",

System.Windows.Forms.MessageBoxButtons.OK,

System.Windows.Forms.MessageBoxIcon.Error);

}

}

else

{

\_showMessage.Show(

"Не все поля заполнены",

"AutoService",

System.Windows.Forms.MessageBoxButtons.OK,

System.Windows.Forms.MessageBoxIcon.Information);

}

}, (o) => CustomerCars.Count > 0));

}

Данные при регистрации проходят валидацию. Валидация реализована с использованием регулярных выражений. Пример валидации приведен в листинге 4.4.

public class RegisterObject : NotifyDataErrorInfo<RegisterObject>

{

private string \_name;

private string \_surname;

private string \_email = string.Empty;

private string \_password= string.Empty;

static RegisterObject()

{

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Name),

"Необходимо заполнить",

x => !string.IsNullOrEmpty(x.Name)));

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Surname),

"Необходимо заполнить",

x => !string.IsNullOrEmpty(x.Surname)));

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Email),

"Необходимо заполнить",

x => !string.IsNullOrEmpty(x.Email)));

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Email),

"Не соответствует шаблону",

x => Regex.IsMatch(x.Email, @"^([a-zA-Z0-9\_-]+\.)\*[a-zA-Z0-9\_-]+@[a-z0-9\_-]+(\.[a-z0-9\_-]+)\*\.[a-z]{2,6}$")));

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Password),

"Необходимо заполнить",

x => !string.IsNullOrEmpty(x.Password)));

Rules.Add(new DelegateRule<RegisterObject>(

nameof(Password),

"Только латиница и цифры",

x => Regex.IsMatch(x.Password, @"^[a-zA-Z0-9]{8,16}$")));

}

}

Листинг 4.4 – Валидация при регистрации

При авторизации проверяется существует ли пользователь с такими данными, а затем проверяется значения хэш-функций паролей. В случае неудовлетворения требований появляется надпись с ошибкой. Пример реализации авторизации приведет в листинге 4.5.

Листинг 4.5 – Реализация авторизации

public RelayCommand LoginCommand

{

get

{

return \_loginCommand ?? (\_loginCommand = new RelayCommand((o) =>

{

var passwordBox = o as PasswordBox;

Login.Password = passwordBox.Password;

var customer = \_unitOfWork.CustomerRepository.GetEntityByConditionOrDefault(c => string.Equals(c.Email, Login.Login));

if (customer != null)

{

if (BCrypt.Net.BCrypt.Verify(Login.Password, customer.Password))

{

CurrentUser.Id = customer.Id;

var main = new MainWindow();

main.Show();

Close();

return;

}

}

var admin = \_unitOfWork.AdminRepository.GetEntityByConditionOrDefault(c => string.Equals(c.Login, Login.Login));

if (admin != null)

{

if (BCrypt.Net.BCrypt.Verify(Login.Password, admin.Password))

{

var main = new MainAdmin();

main.Show();

Close();

return;

}

}

\_showMessage.Show(

"Логин или пароль введены неверно!",

"AutoService",

System.Windows.Forms.MessageBoxButtons.OK,

System.Windows.Forms.MessageBoxIcon.Information);

}));



## **Создание и редактирование заказа**

Создание заказа осуществляется с помощью паттерна Command. Новый заказ добавляется в репозиторий заказов OrderRepository. После чего добавляется новое уведомление Notification в NotificationRepository, уведомляя пользователя о создании заказа. Листинг команды создания заказа приведен ниже.

public RelayCommand MakeCommand

{

get

{

return \_makeCommand ?? (\_makeCommand = new RelayCommand((o) =>

{

var newOrder = new Order

{

Id = Order.Id,

CarId = Order.CustomerCar.Id,

Sum = Order.Sum,

CreatedAt = DateTime.Now,

EmployeeId = Order.EmpId,

OrderStatusId = Models.Enums.OrderStatuses.Accepted,

StartDate = Order.StartDate,

Services = Order.Services

};

\_vm.UnitOfWork.OrderRepository.Add(newOrder);

var notification = new Notification

{

CreatedAt = DateTime.Now,

CustomerId = Order.CustomerCar.CustomerId,

Message = $"Заказ принят. Номер {newOrder.Id}"

};

\_vm.UnitOfWork.NotificationsRepository.Add(notification);

\_vm.UnitOfWork.SaveChanges();

\_vm.ShowMessage.Show(

"Оформлен",

"AutoService",

System.Windows.Forms.MessageBoxButtons.OK,

System.Windows.Forms.MessageBoxIcon.Information);

\_orders.AllOrders.Add(newOrder);

\_orders.Orders.Add(newOrder);

\_vm.Current = \_orders;

}, (o) => !Order.HasErrors));

}

Листинг 4.6 – Команда создания заказа

Редактирование заказа также реализуется с помощью паттерна Command (листинг 4.7).

public RelayCommand ChangeCommand

{

get

{

return \_changeCommand ?? (\_changeCommand = new RelayCommand((o) =>

{

\_orders.SelectedOrder.Car = Order.CustomerCar;

\_orders.SelectedOrder.Sum = Order.Sum;

\_orders.SelectedOrder.EmployeeId = Order.EmpId;

\_orders.SelectedOrder.OrderStatus = Order.Status;

\_orders.SelectedOrder.StartDate = Order.StartDate;

\_orders.SelectedOrder.Services = Order.Services;

\_vm.UnitOfWork.OrderRepository.Update(\_orders.SelectedOrder);

var notification = new Notification

{

CreatedAt = DateTime.Now,

CustomerId = Order.CustomerCar.CustomerId,

Message = $"В заказ под номером {Order.Id} были внесены изменения"

};

if(Order.Status.Id == OrderStatuses.Processing)

{

notification.Message = $"Выполняется заказ под номером {Order.Id}";

}

else if(Order.Status.Id == OrderStatuses.Canceled)

{

notification.Message = $"Заказ под номером {Order.Id} был отменен";

}

else if(Order.Status.Id == OrderStatuses.Fulfilled)

{

notification.Message = $"Заказ под номером {Order.Id} выполнен";

}

\_vm.UnitOfWork.NotificationsRepository.Add(notification);

\_vm.UnitOfWork.SaveChanges();

\_vm.ShowMessage.Show(

"Изменен",

"AutoService",

System.Windows.Forms.MessageBoxButtons.OK,

System.Windows.Forms.MessageBoxIcon.Information);

\_orders.SelectedOrder = null;

\_vm.Current = \_orders;

}, (o) => !Order.HasErrors && CanEdit));

}

Листинг 4.7 – Реализация редактирования заказа



## **Создание отзывов**

Создание отзывов осуществляется по той же схеме, что и создание заказов – с помощью Command (листинг 4.8).

public RelayCommand AddCommand

{

get

{

return \_addCommand ?? (\_addCommand = new RelayCommand((o) =>

{

if(!string.IsNullOrEmpty(AddReview.Title) && !string.IsNullOrEmpty(AddReview.Description))

{

Review newReview = new Review

{

Id = AddReview.Id,

Title = AddReview.Title,

Description = AddReview.Description,

CreatedAt = AddReview.CreatedAt,

CustomerId = \_vm.CurrentCustomer.Id

};

\_vm.UnitOfWork.ReviewRepository.Add(newReview);

\_vm.UnitOfWork.SaveChanges();

Reviews.Add(AddReview);

AddReview = new ReviewObject

{

Customer = \_vm.UnitOfWork.CustomerRepository.GetEntityByConditionOrDefault(c => c.Id == \_vm.CurrentCustomer.Id)

};

}

}));

Листинг 4.8 – Реализация создания отзывов



# **Тестирование и проверка работоспособности**

## **5.1 Тестирование регистрации и авторизации**

В момент авторизации и регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, неверный пароль, некорректный адрес электронной почты. Такие исключения обрабатываются программным средством с помощью показа пользователю всплывающих сообщений с текстом ошибки. Примеры обработки разных видов исключительных ситуаций представлены на рисунках 5.1 – 5.3.

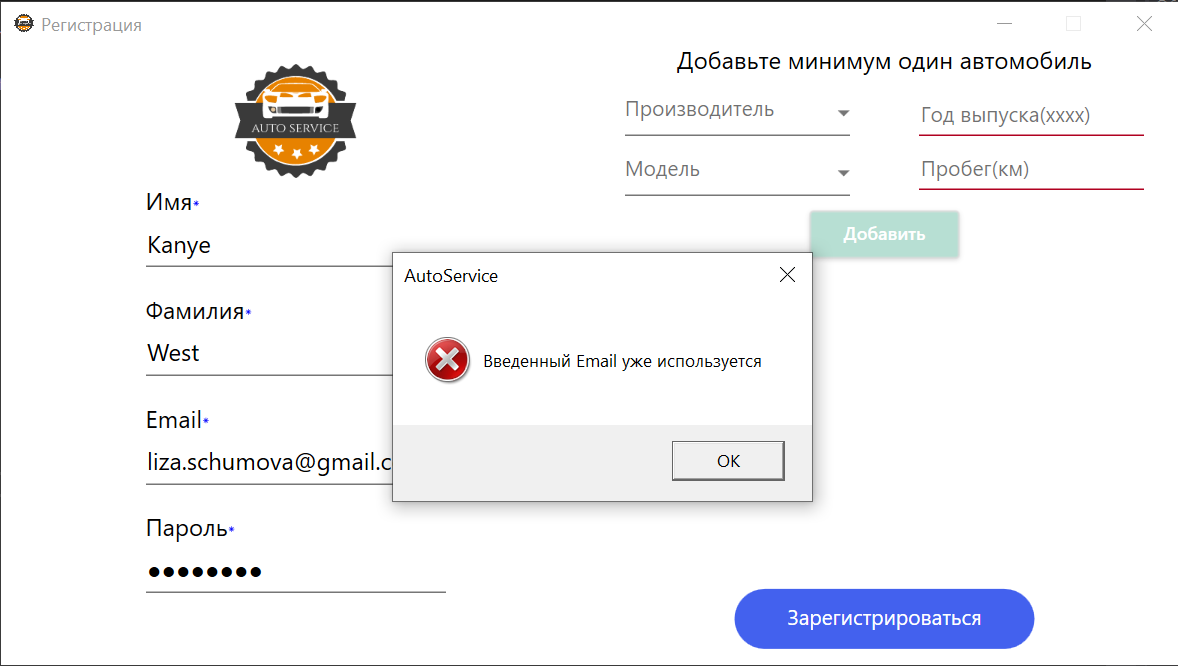


Рисунок 5.1 – Обработка существующего адреса электронной почты

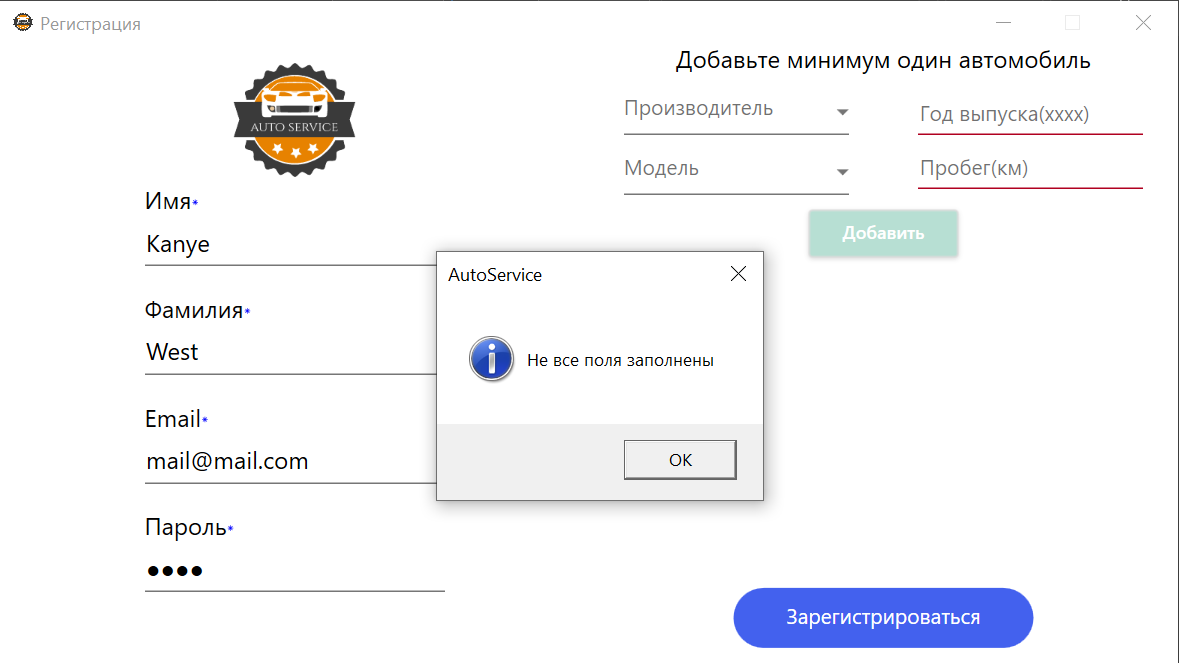


Рисунок 5.2 – Обработка незаполненных полей

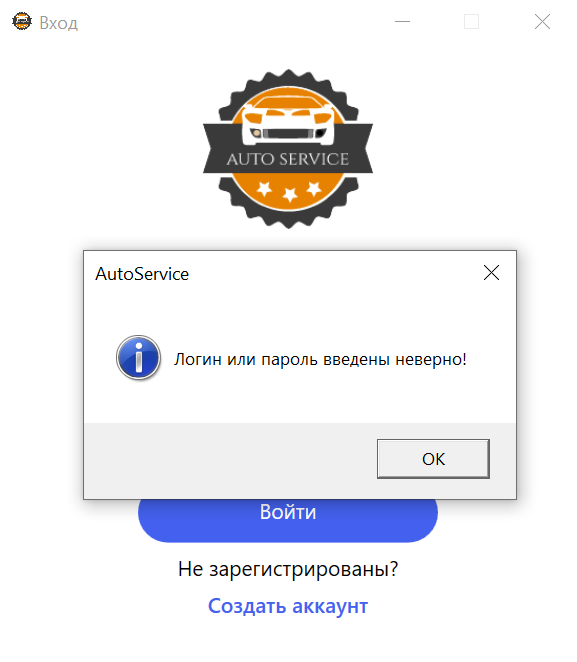


Рисунок 5.3 – Обработка некорректного ввода значений

## **Тестирование добавления объектов базы данных**

При добавлении новых сотрудников и услуг незаполненные поля подчеркиваются красным цветом, а кнопка добавления становится недоступной, пока не заполнятся все поля (рисунок 5.4).

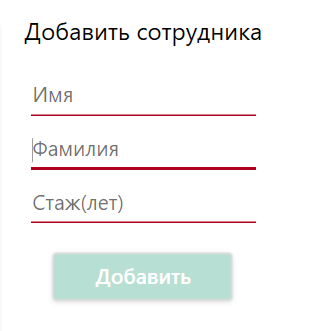


Рисунок 5.4 – Пустые поля на формах подчеркиваются красным



# **Руководство по использованию**

При запуске программного средства пользователь попадает на страницу авторизации, содержащая формы входа и регистрации.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, а не зарегистрированному – перейти на страницу регистрации нажатием на кнопку «Создать аккаунт» и зарегистрироваться в системе, после чего пользователь будет перенаправлен на страницу авторизации, с которой он может совершить вход в систему.

Страницы авторизации и регистрации представлены на рисунках 6.1 и 6.2.

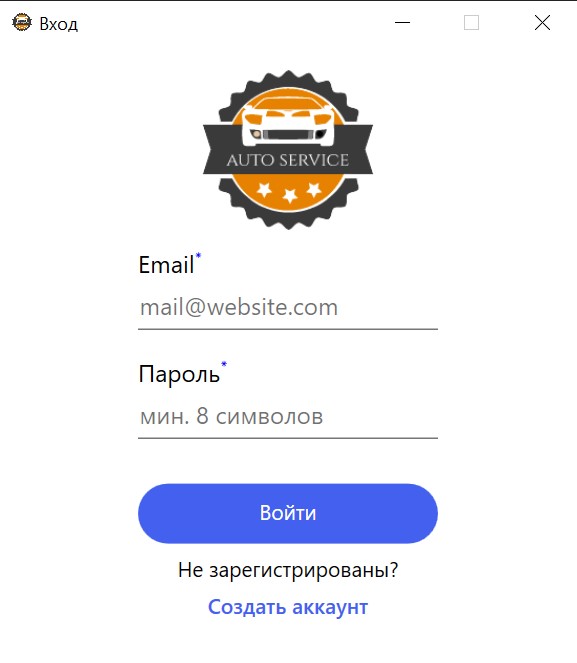


Рисунок 6.1 – Авторизация

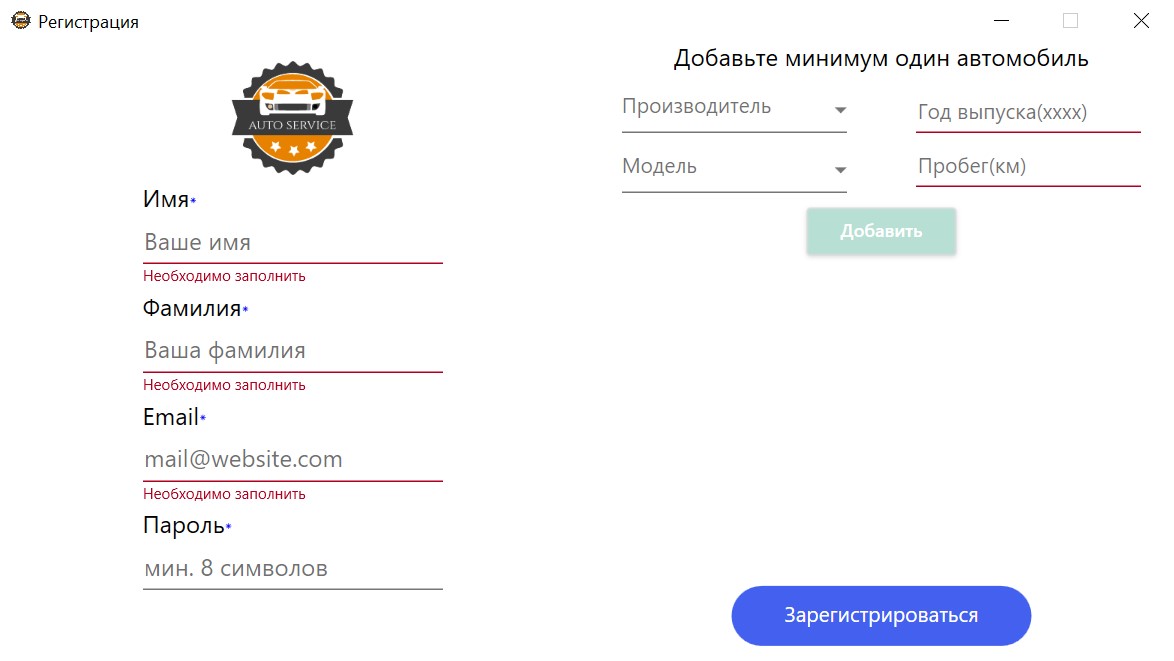


Рисунок 6.2 – Регистрация

После авторизации пользователь попадает на главную страницу приложения. Здесь он может просмотреть уведомления о состояниях заказа, добавить новые автомобили, изменить личные данные (рисунок 6.3).

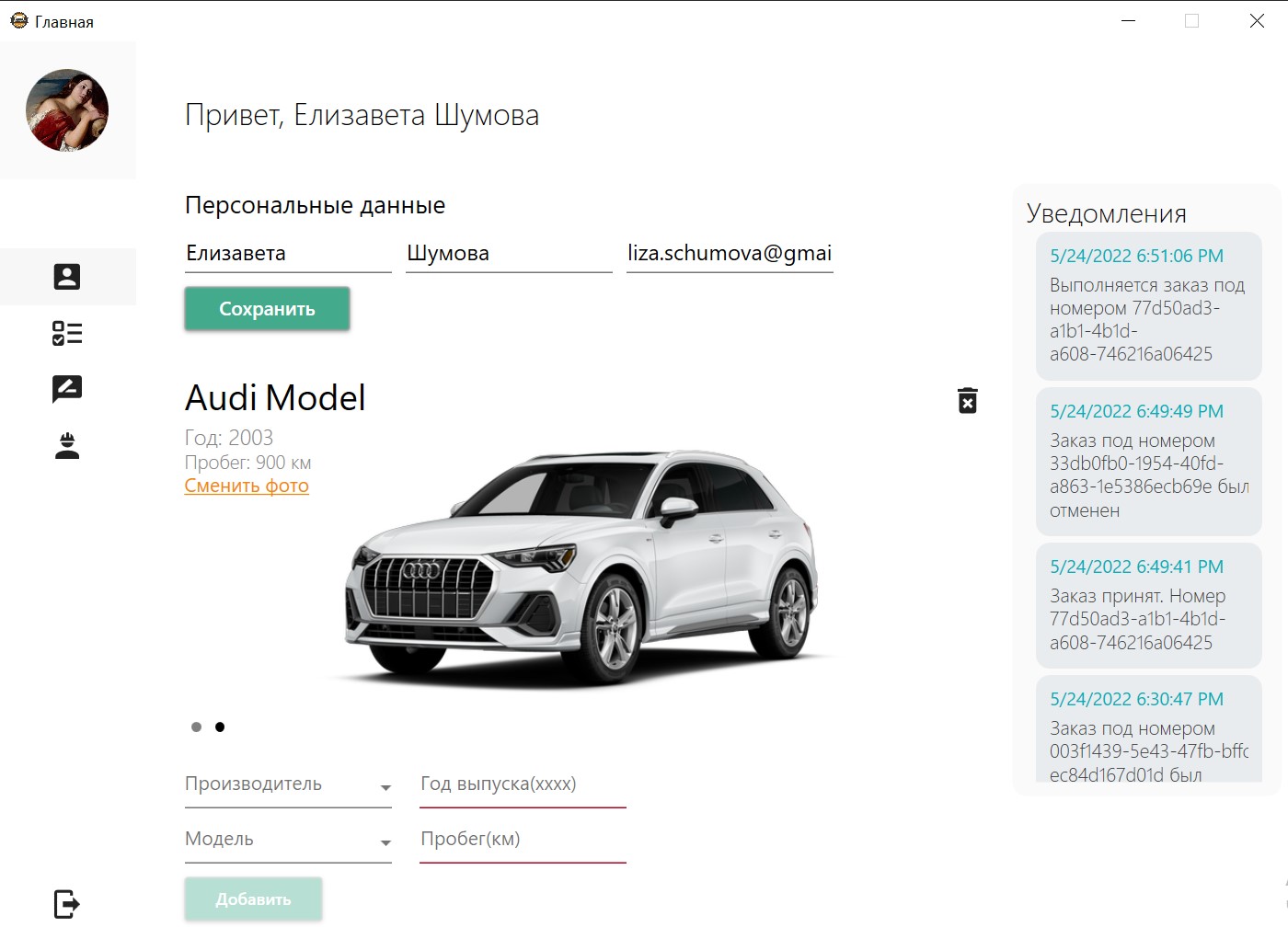


Рисунок 6.3 – Главная страница пользователя

С главной страницы пользователь может перейти на вкладку заказов в боковом меню. Здесь он может оформить новый заказ либо провести поиск по истории заказов. Окно заказов и форма заказа представлены на рисунках 6.4 и 6.5.

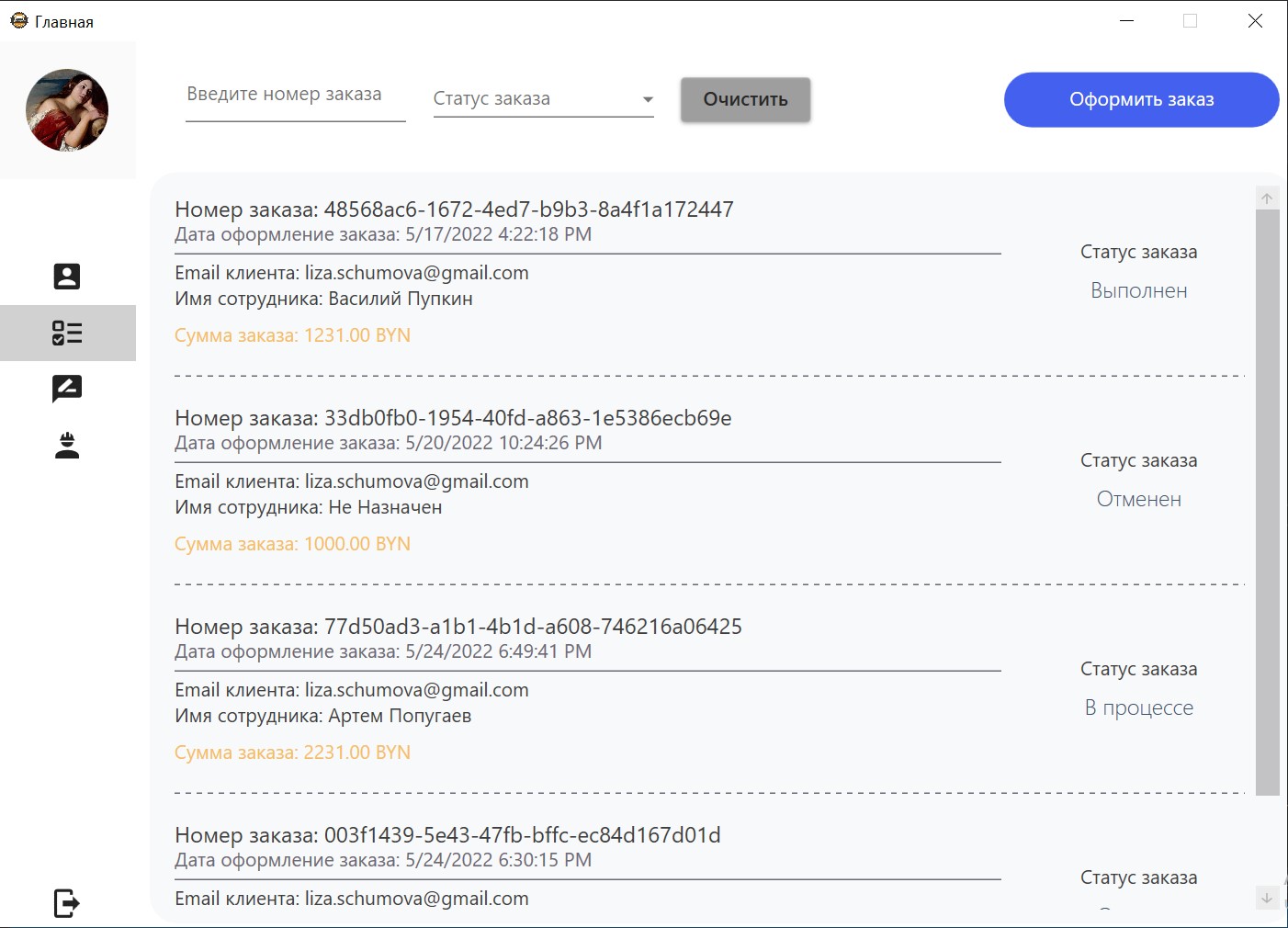


Рисунок 6.4 – Окно заказов

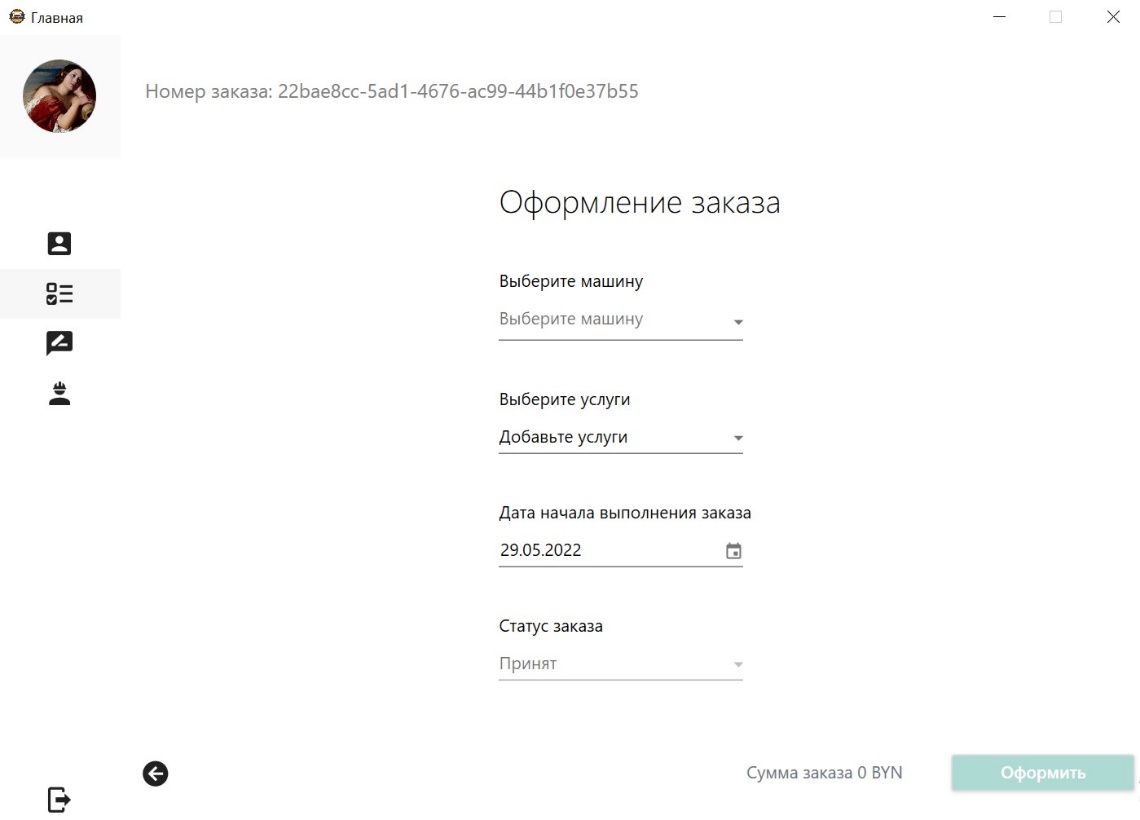


Рисунок 6.4 – Форма заказа

Также пользователь может оставить отзыв в специальном окне (рисунок 6.5).

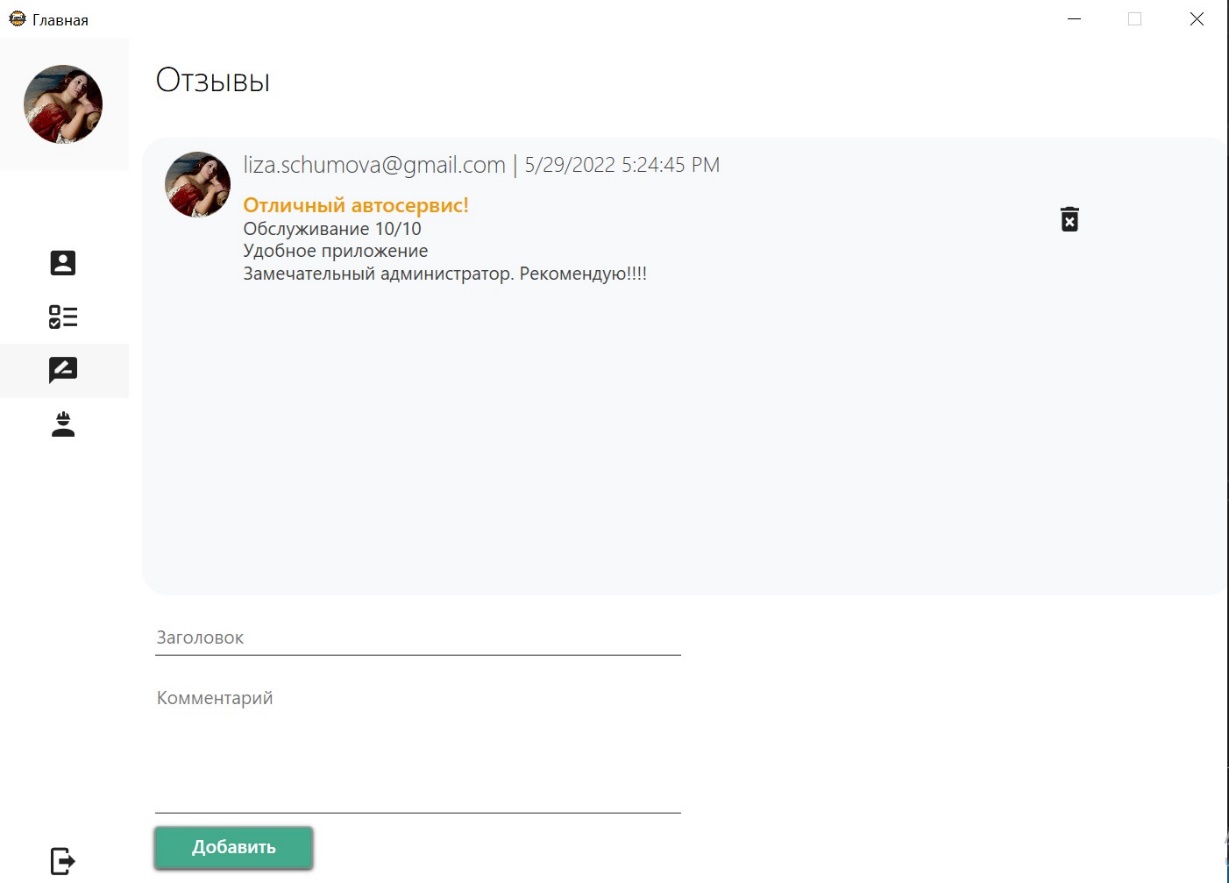


Рисунок 6.5 – Отзывы

Пользователь может просмотреть список сотрудников автосервиса, как на рисунке 6.6.

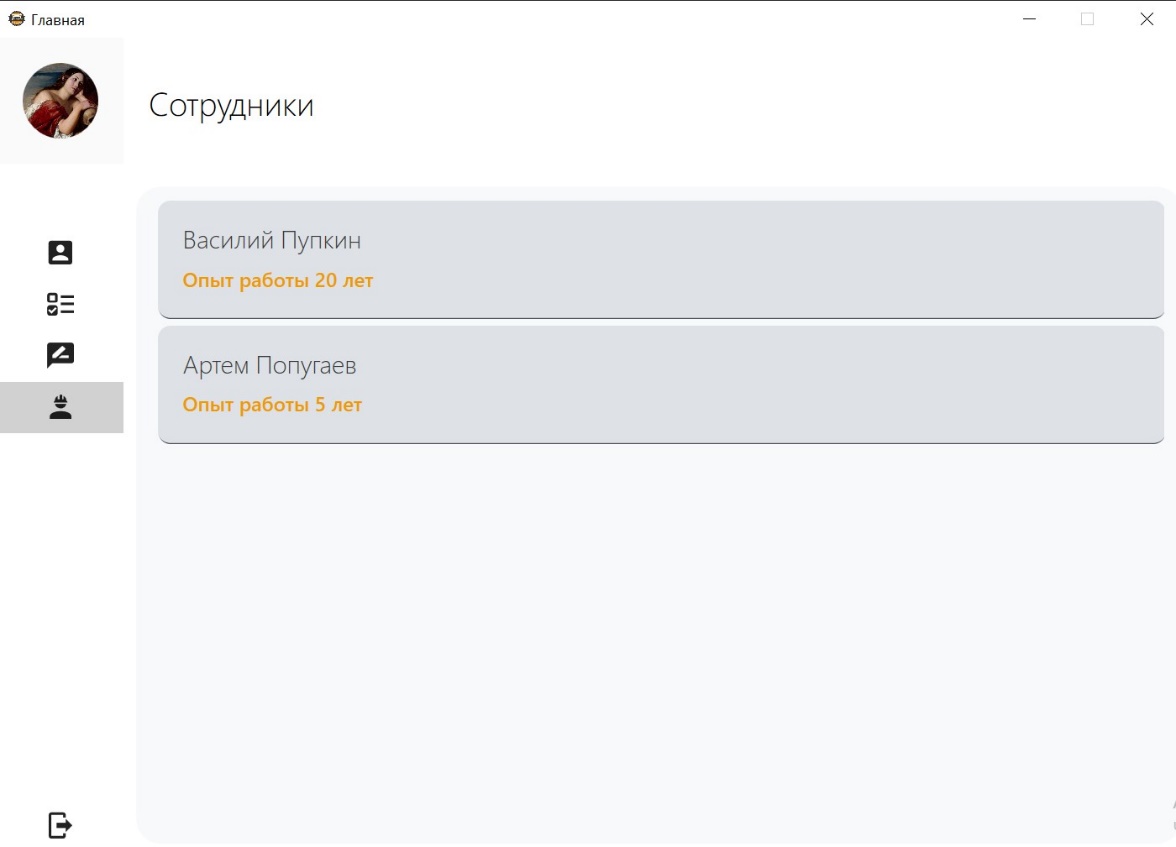


Рисунок 6.6 – Сотрудники

Окна администратора приведены на рисунках 6.7 – 6.13.

Администратор может добавлять производителей и модели автомобилей, обслуживание которых проводит автосервис.

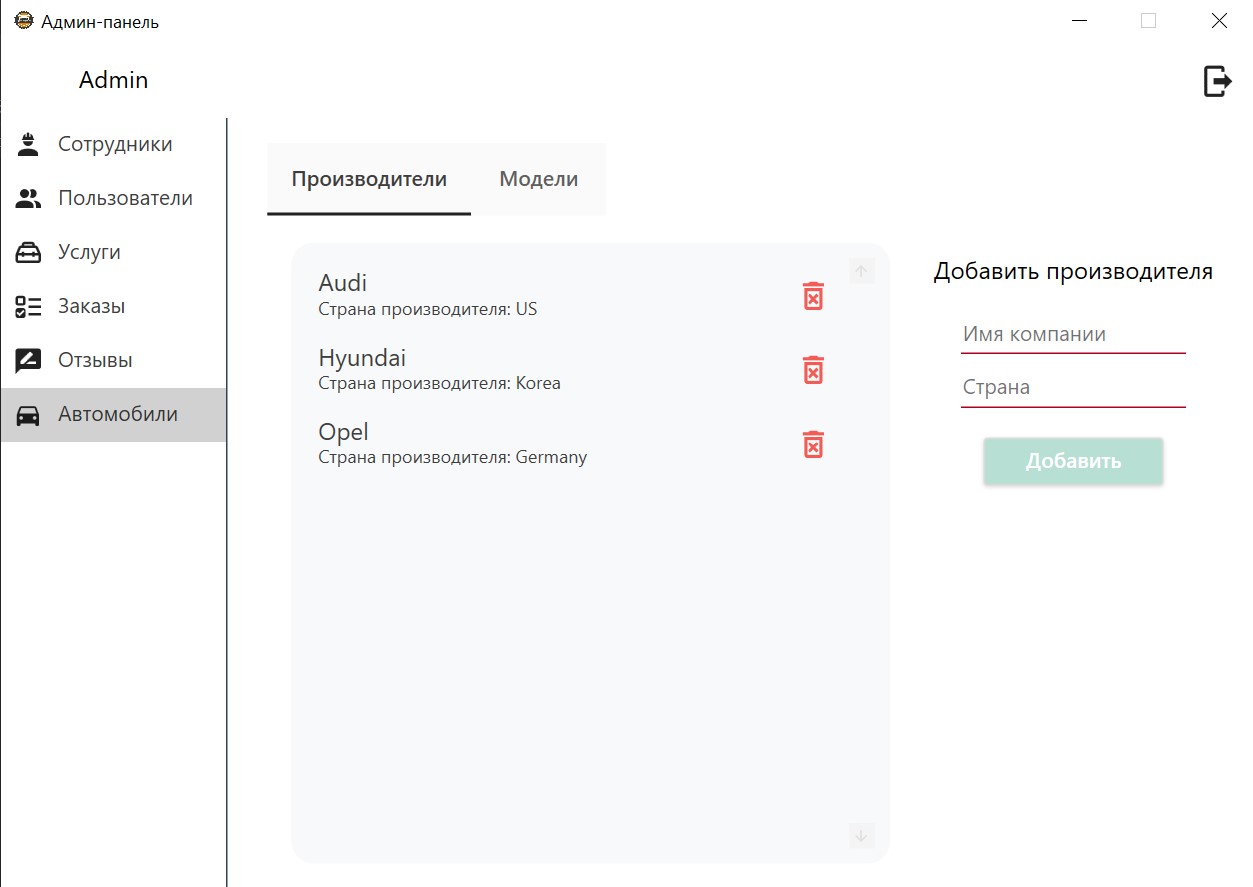


Рисунок 6.7 – Автомобили

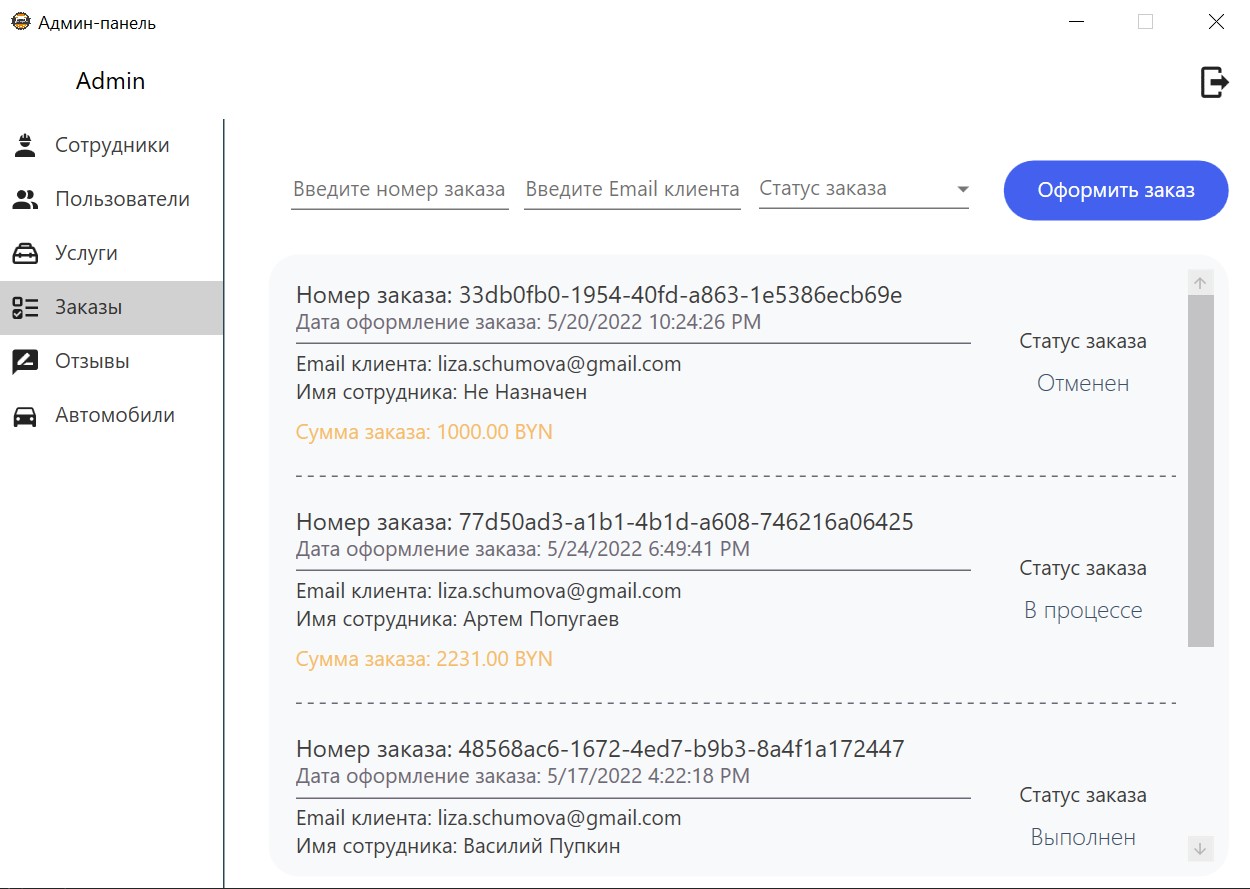


Рисунок 6.8 – Заказы администратора

Администратор имеет право удалять отзывы других пользователей.

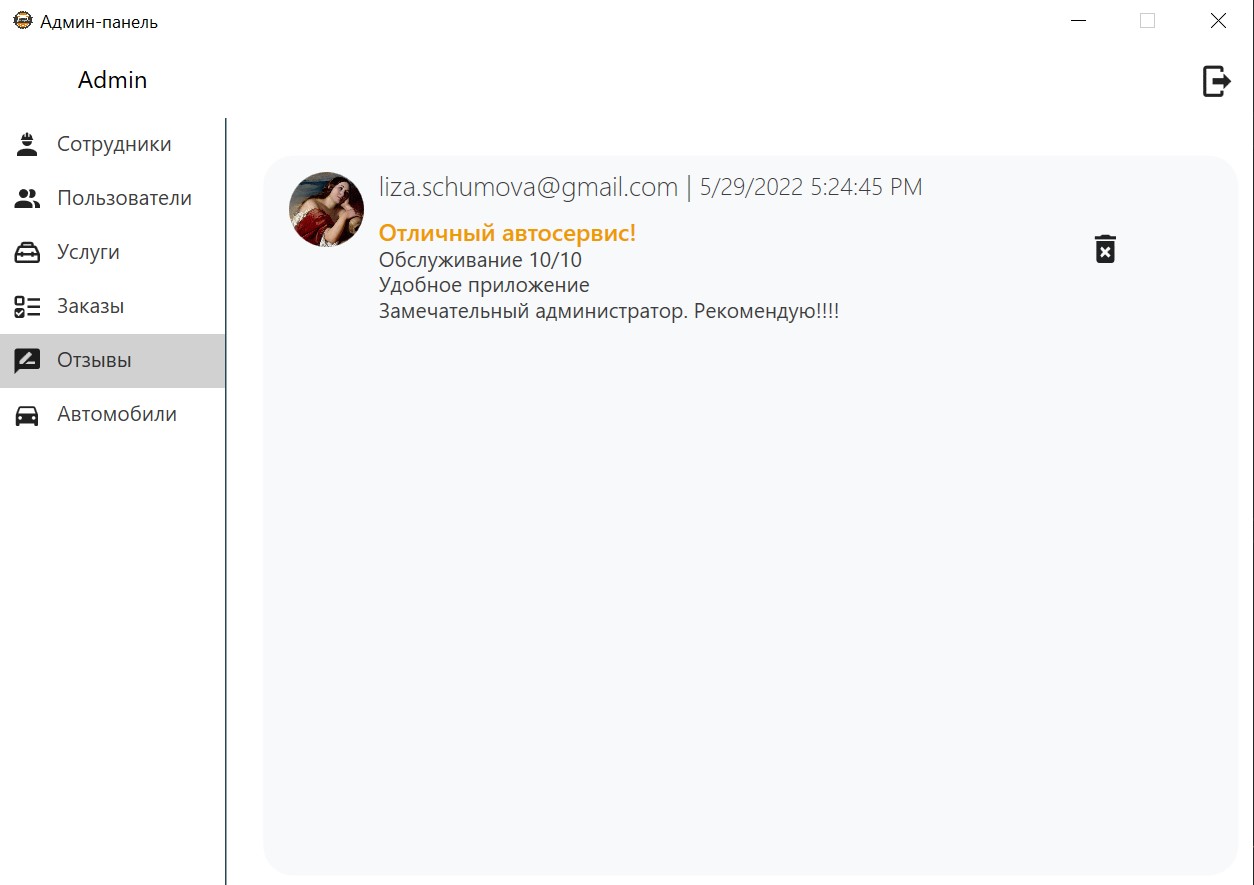


Рисунок 6.9 – Отзывы администратора

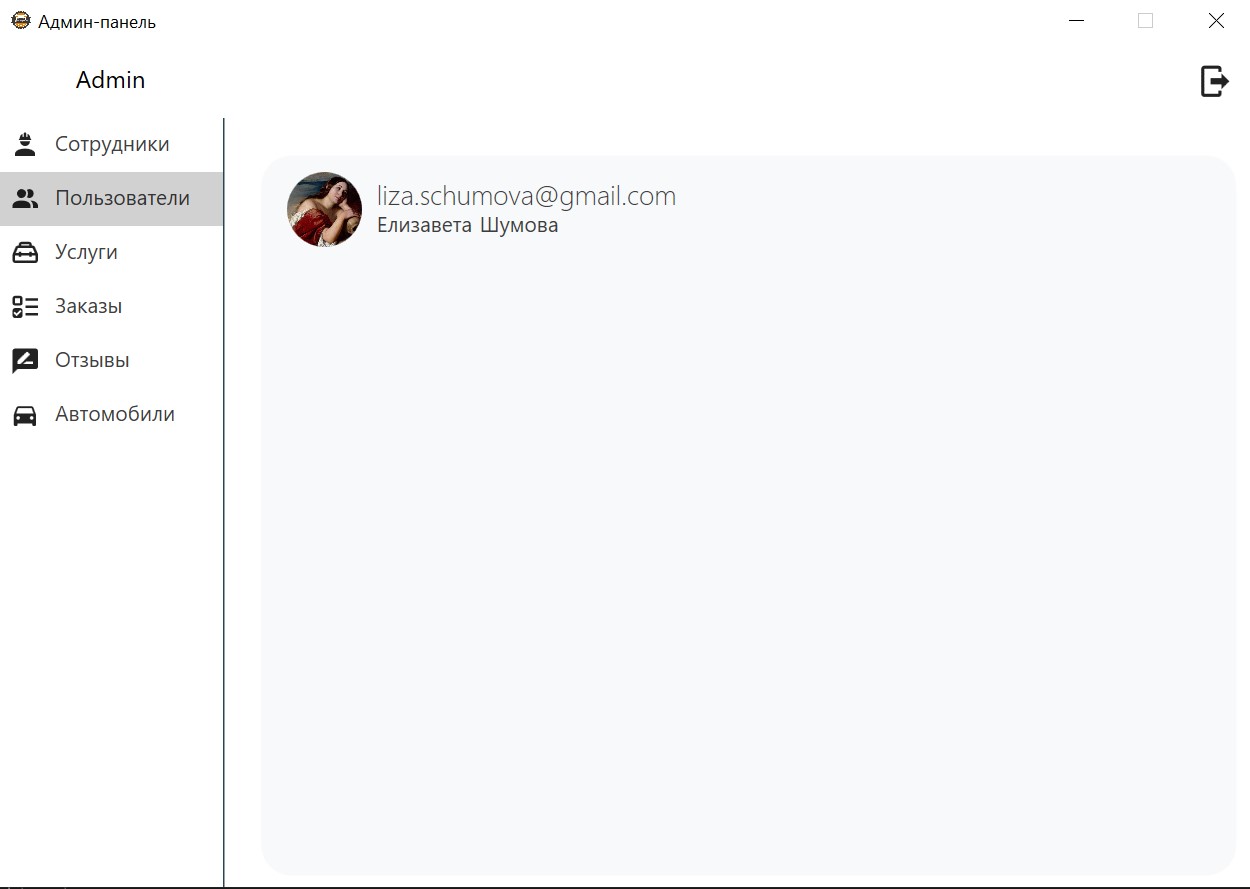


Рисунок 6.10 – Пользователи администратора

Администратор может добавлять и удалять сотрудников.

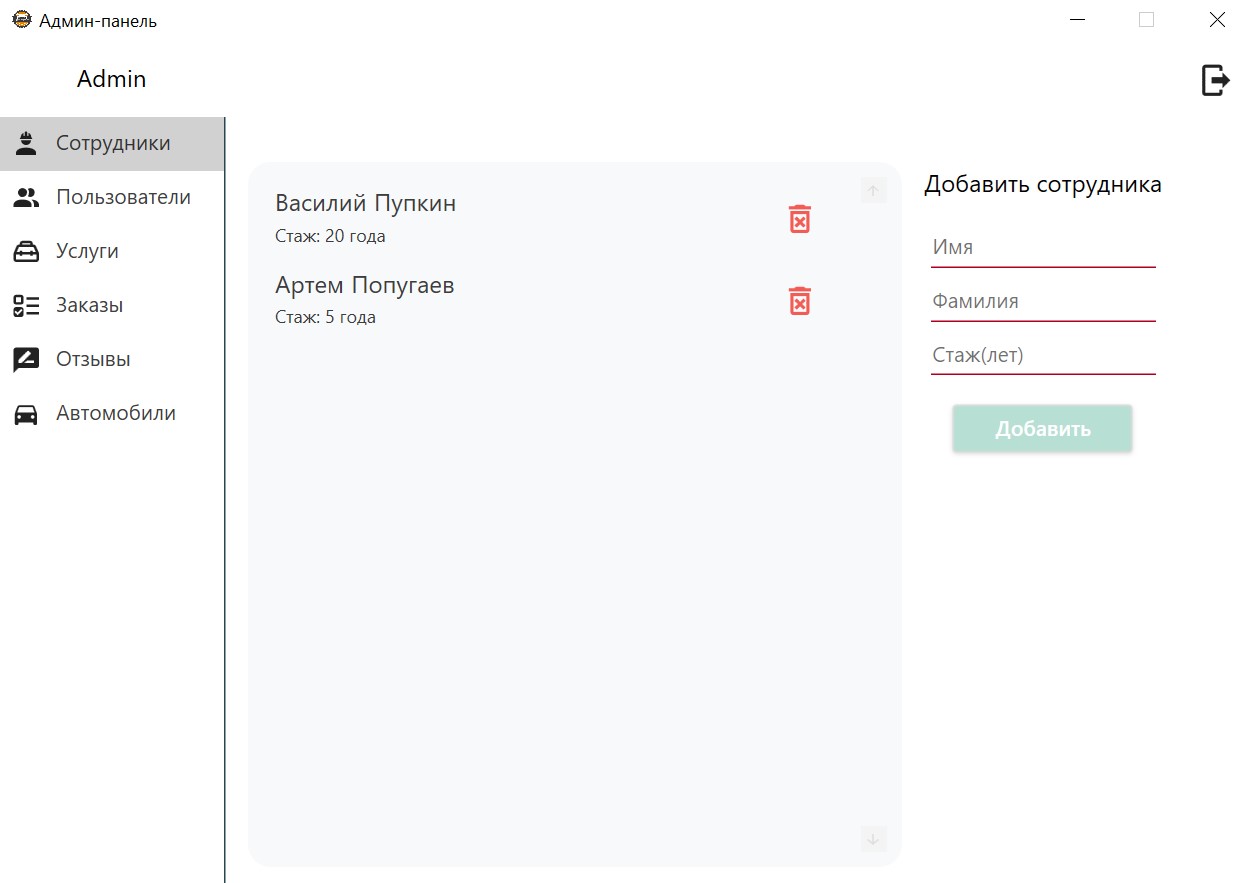


Рисунок 6.11 – Сотрудники администратора

Администратор может добавлять и удалять услуги, которые предоставляет автосервис.

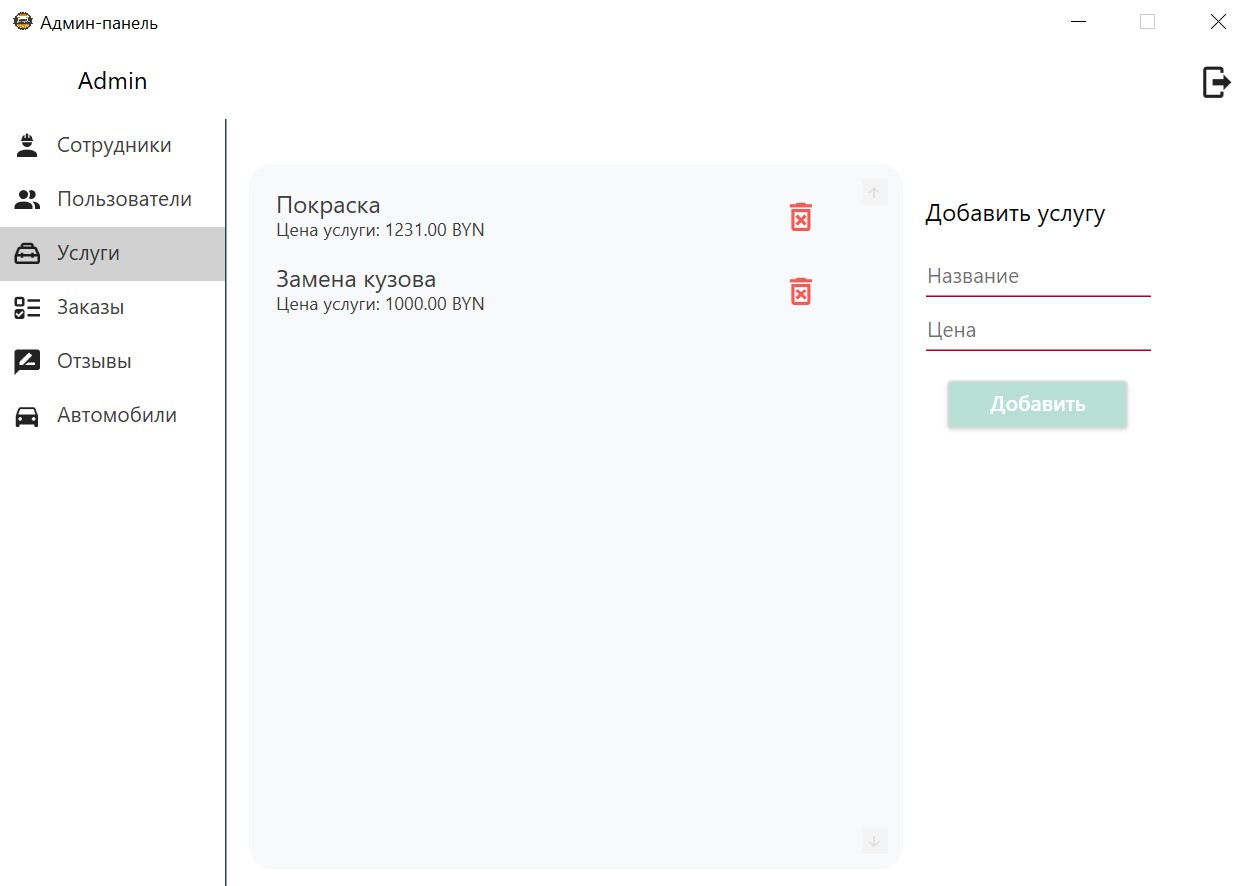


Рисунок 6.12 – Услуги администратора

Администратор может назначать сотрудников, ответственных за заказ, а также изменять статус заказа.

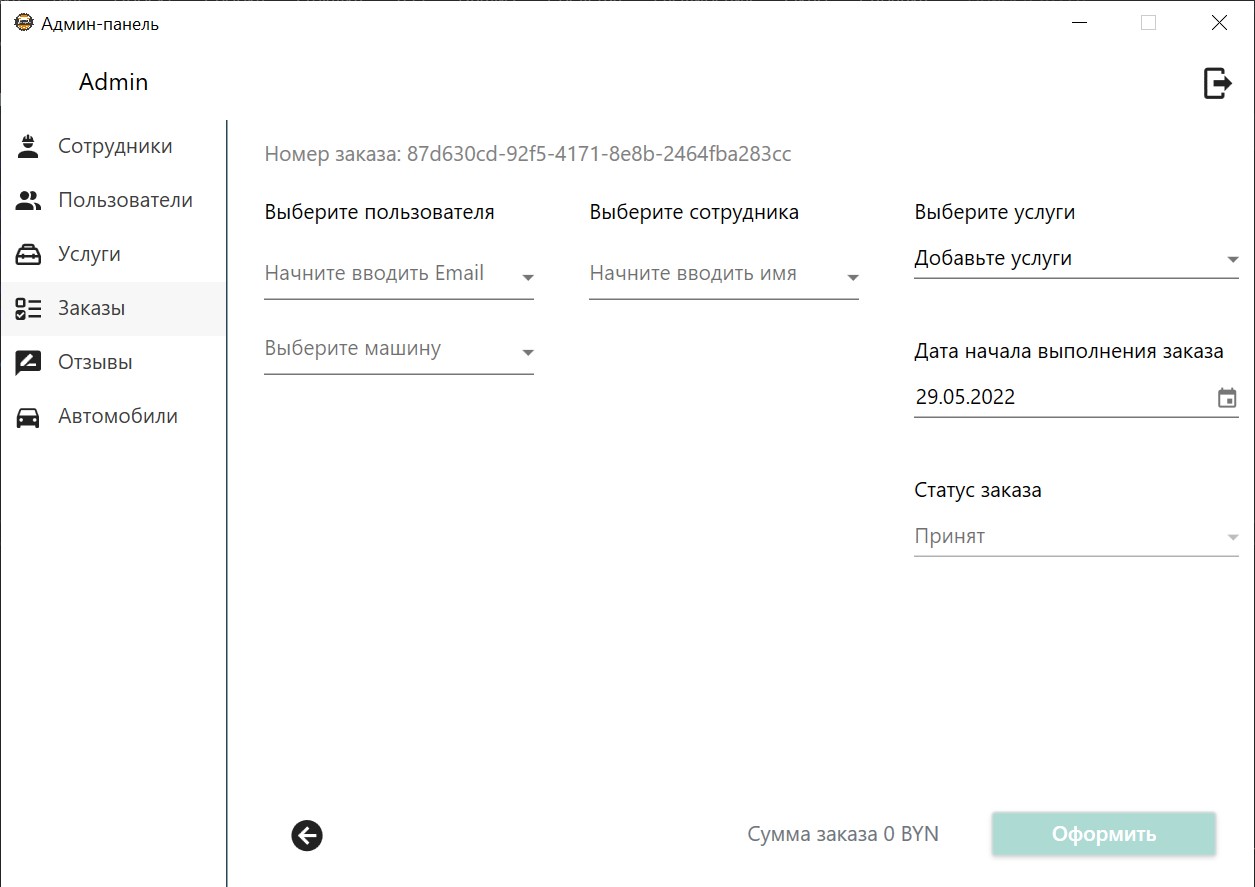


Рисунок 6.13 – Форма заказа администратора

# **Заключение**

В ходе выполнения курсового проекта было разработано программное средство «Repair Auto Services» на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке программного средства были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

* создана база данных;
* реализована возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* реализована возможность авторизированным пользователям оформления заказа;
* реализована возможность работы с объектами базы данных администратором;
* реализована возможность оставлять отзывы;
* реализовано уведомление пользователя о заказе.

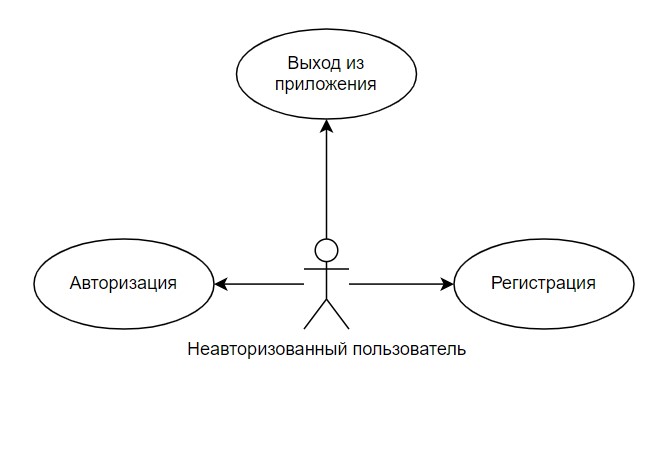
Тестирование программы показало, что она работает корректно и выполняет все свои функции.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

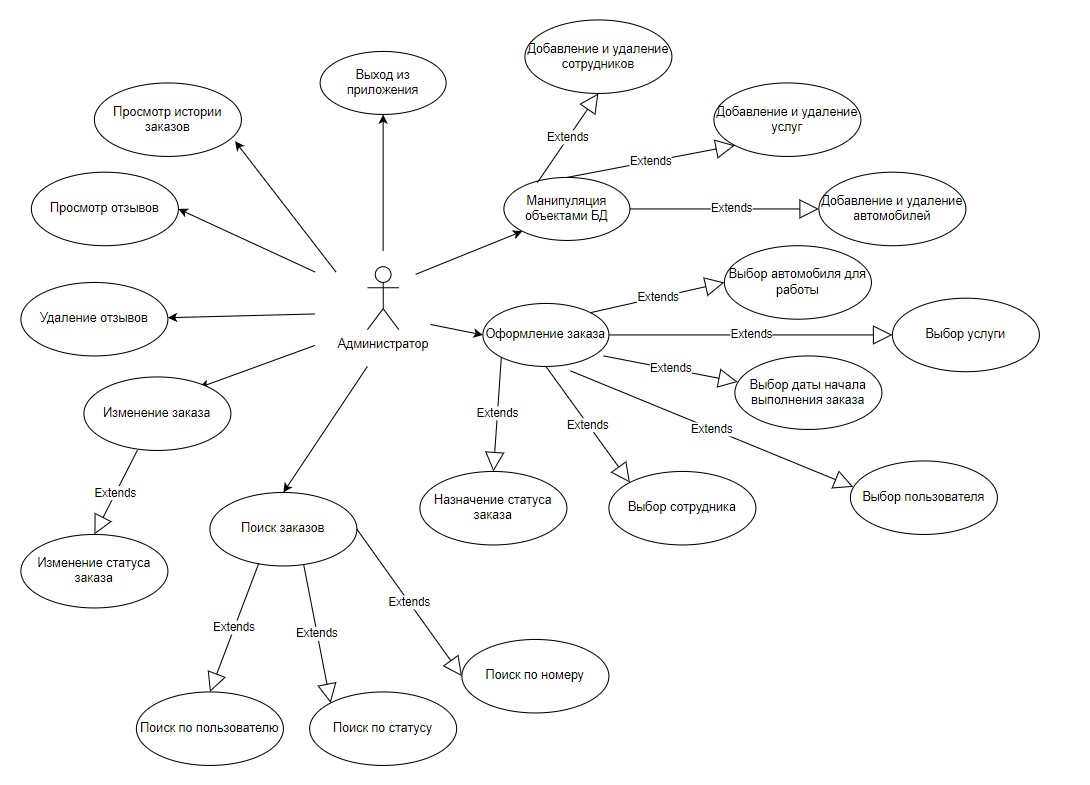
# **Список литературы**

1. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.02.2022
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 14.03.2022
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.03.2022
5. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 03.05.2022
6. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

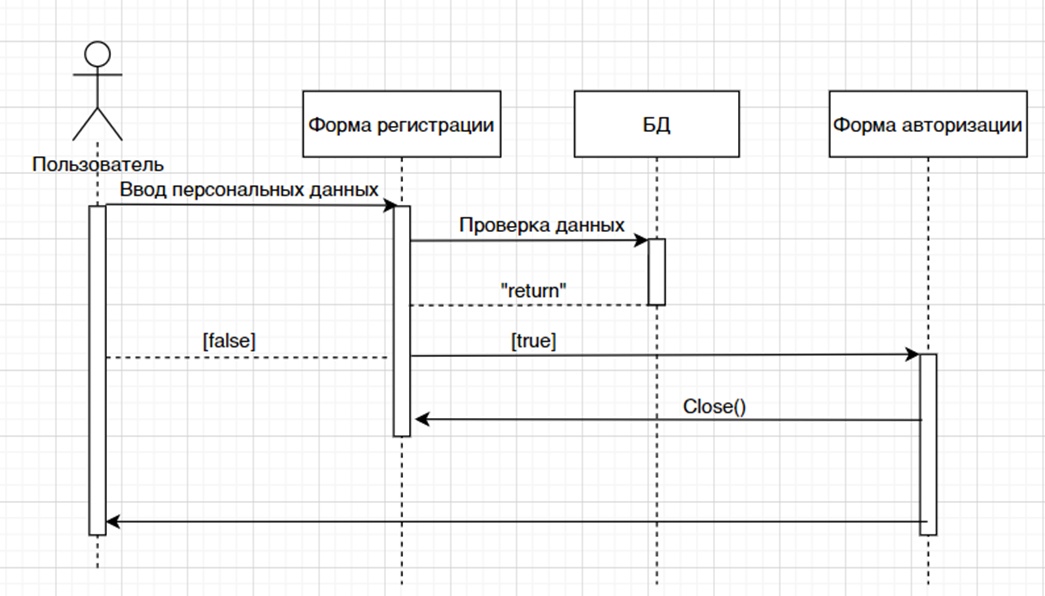
# **Приложение А**



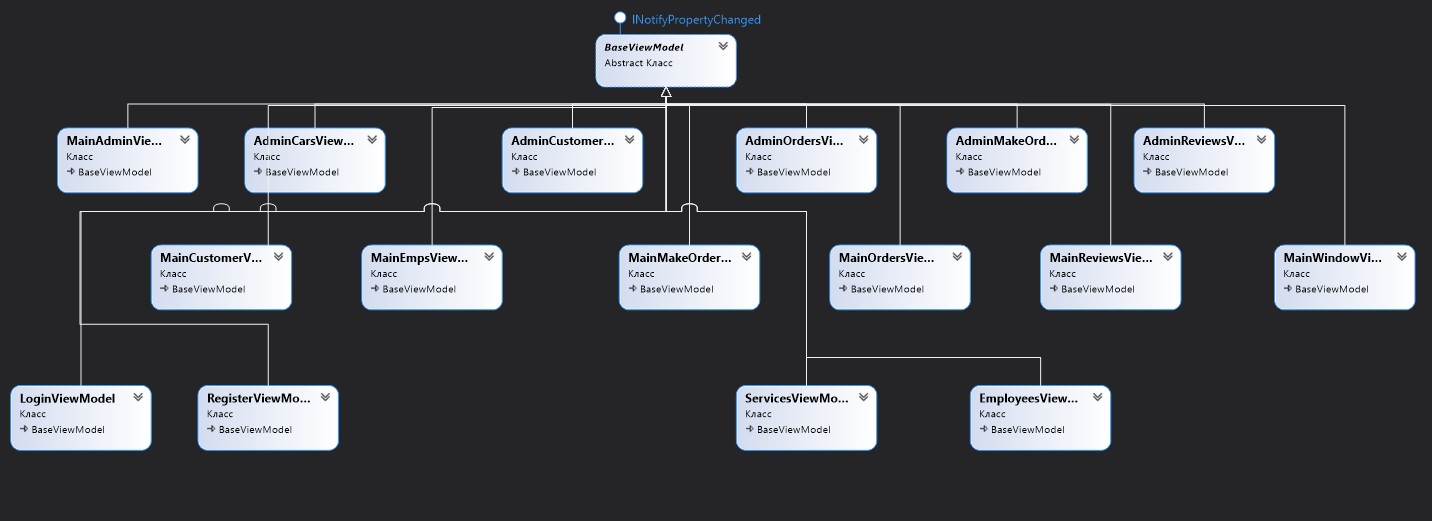




# **Приложение Б**



# **Приложение В**



# **Приложение Г**

