Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

КУРСОВАЯ РАБОТА

По Распределенные системы обработки данных

Тема Распределённая ИС «Детейлинг центр»

Руководитель

Белякова А.С

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Кокурин Я.Д.

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

В данной курсовой работе необходимо было спроектировать информационную систему детейлинг центра. В качестве средств разработки приложения была использована среда Microsoft Visual Studio 2022. Язык разработки: С#. Технология ASP.NET.

In this course work, it was necessary to design an information system for a detailing center. The Microsoft Visual Studio 2022 environment was used as the application development tools. Development language: C#. Technology ASP.NET

Содержание

[Введение 6](#_Toc167387479)

[1 Анализ технического задания 8](#_Toc167387480)

[2 Разработка моделей данных 12](#_Toc167387481)

[2.1 Концептуальная модель данных 12](#_Toc167387482)

[2.2 Логическая модель данных 12](#_Toc167387483)

[2.3 Физическая модель данных 13](#_Toc167387484)

[3 Проектирование работы системы 16](#_Toc167387485)

[4 Разработка и реализация системы 18](#_Toc167387486)

[5 Тестирование системы 24](#_Toc167387487)

[Заключение 26](#_Toc167387488)

[Список литературы 27](#_Toc167387489)

[Приложение 1 28](#_Toc167387490)

[Приложение 2 30](#_Toc167387491)

[Приложение 3 31](#_Toc167387492)

Введение

В настоящее время для достижения успехов любому предприятию необходимо точно понимать свои затраты, прибыли, ресурсы, бизнес-процессы и многое другое. Наглядная информация о происходящем поможет глубже проанализировать процесс и сделать правильные выводы, что в конечном итоге приведет к повышению эффективности.

Лидерами становятся наиболее эффективные предприятия, имеющие минимальные издержки, высочайший уровень производительности и полностью контролируемые и четко отлаженные процессы. Ни что так не способствует контролю и анализу деятельности на предприятии как внедрение информационной системы – программно-аппаратный комплекс для авторизации деятельности организации, который обеспечивает хранение, передачу и обработку информации.

Внедрив информационную систему, руководство предприятия сможет принимать правильные решения по повышению эффективности отдельных процессов, снизит затраты, улучшит коммуникацию что несомненно поспособствует достижению успехов в своем виде деятельности.

Распределенные системы обработки данных являются неотъемлемой частью современных информационных технологий. Они позволяют эффективно обрабатывать большие объемы информации, распределять вычислительные ресурсы и обеспечивать высокую доступность данных.

В данной курсовой работе рассматривается создание распределенной информационной системы для автоматизации предметной области детейлинг центра на основе технологии ASP.NET Core MVC.

Детейлинг центр – предприятие, занимающееся оказанием специализированных процедур по уходу за автомобилем. Детейлинг подразумевает максимально тщательную и качественную проработку внешнего и внутреннего вида автомобиля до состояния близкого к идеальному и сохранение этого результата на длительный период времени. Для повышения эффективности деятельности детейлинг центра требуется анализировать спрос клиентов на услуги предприятия, для этого необходимы отчёты о востребованности услуг, на основании которых будут приниматься дальнейшие решения руководством (об изменении списка/качества оказываемых услуг).

Целями данной работы являются разработка моделей информационной системы детейлинг центра с учётом требуемых обработки данных для повышения эффективности предприятия и проектирование информационной системы.

Задачей проекта является программная реализация информационной системы, проверка её работоспособности и получения необходимых отчётов.

1. Анализ технического задания

В Данной курсовой работе необходимо создать информационную систему детейлинг центра.

Для разработки такой системы необходимы СУБД для создания базы данных и среда разработки для создания интерфейса и функций по работе с БД.

Разрабатываемое приложение предназначено для использования в детейлинг центре, занимающемся оказанием специализированных процедур по уходу за автомобилем.

Реализуемая программа должна разработана на технологии ASP.NET Core MVC. Технология ASP.NET Core MVC является одной из наиболее популярных и мощных платформ для разработки веб-приложений. Она предоставляет разработчикам широкий набор инструментов и возможностей для создания масштабируемых, высокопроизводительных и безопасных приложений. Процесс работы APS.Net MVC представлен на рисунке 1.

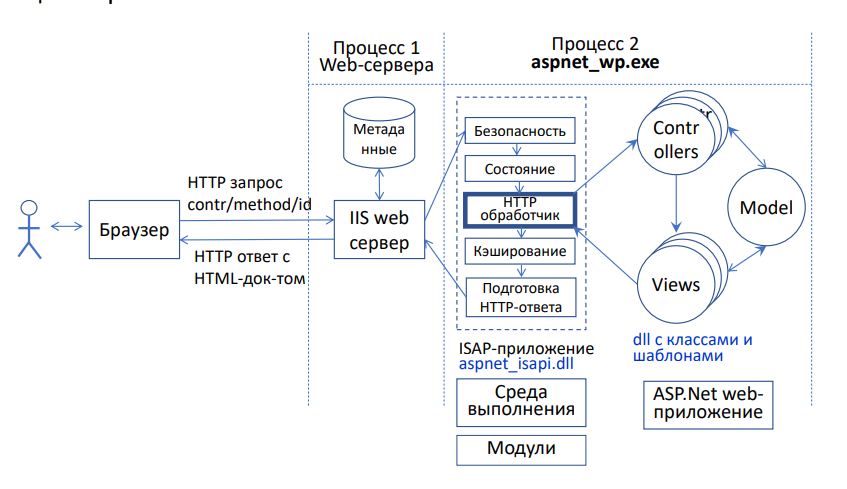


Рисунок 1 – Процесс работы APS.Net MVC

К достоинствам ASP.Net MVC относятся:

1) Разделение ответственности:

- MVC-приложение состоит из трех частей (контроллера, представления модели), каждая из которых выполняет свои специфичные функции (в итоге приложение будет легче поддерживать модифицировать в будущем);

- В силу разделения ответственности приложения MVC обладают лучшей тестируемостью (можно тестировать отдельные компоненты независимо друг от друга).

2) Соответствие протоколу HTTP:

- MVC-приложения в отличие от веб-форм не поддерживают объекты состояния (ViewState);

- Ясность и простота платформы позволяют добиться большего контроля над работой приложения.

3) Гибкость:

- можно настраивать различные компоненты платформы по своему усмотрению;

- можно изменять части конвейера работы MVC или адаптировать его к своим нуждам и потребностям.

Технология ASP.NET MVC – это фреймворк на основе которого разработчик создает свое web-приложение. Фреймворк включает набор совместно работающих классов, а также вспомогательные конструкции. Разработчику для создания приложения нужно понять логику работы фреймворка и способ подключения для выполнения требуемых работ. Подключение выполняется за счет создания производных классов и вызова нужных методов.

Компоненты MVC. Архитектурный шаблон Model-View-Controller (MVC) разделяет web-приложение на части, которые легче кодировать, тестировать и поддерживать.

Модель (model) состоит из файлов, которые содержат код доступа к данным, код бизнес-логики, код валидации данных.

Представление (view) состоит из файлов, которые используются для создания HTML документа, содержащего интерфейс пользователя.

Контроллер (controller) состоит из файлов, которые получают запросы пользователя, требуемые данные из модели, после чего передают эти данные нужному представлению.

Модель представления (view model) – необязательный компонент, который состоит из файлов, единственной работой которых является передача данных между контроллерами и представлениями. Технически это не является частью паттерна MVC, но view models часто используются при работеASP.NET MVC. Схема работы MVC-фреймворка отражена на рисунке 2. Обработка запросов к MVC приложению представлена на рисунке 3.

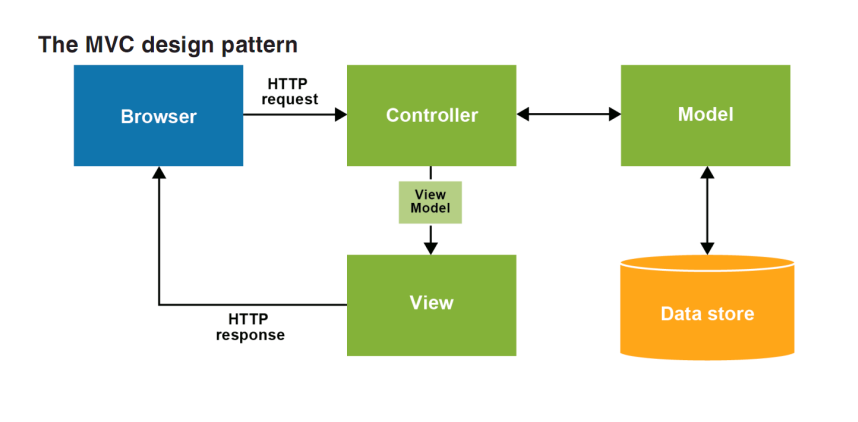


Рисунок 2 – Схема работы MVC-фреймворка

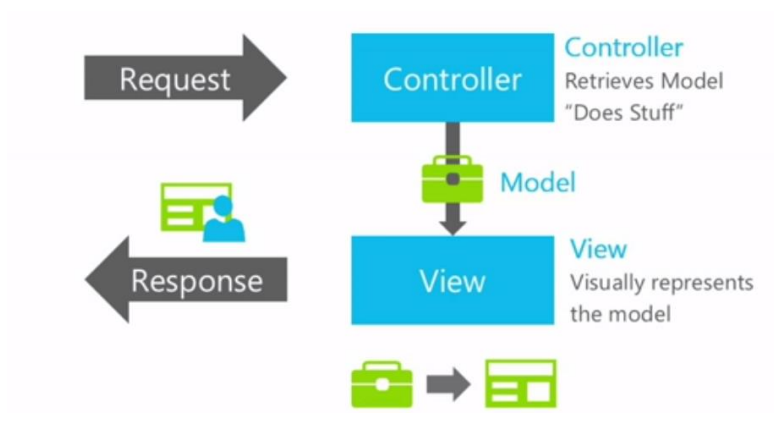


Рисунок 3 – Обработка запросов к MVC приложению

Программный продукт предназначен для автоматизации деятельности детейлинг центра. Администратор реализует CRUD методы обработки данных сотрудников, услуг детейлинг центра. Сотрудник детейлинг-центра управляет записью клиентов на обслуживание, реализует CRUD методы обработки данных клиентов и их автомобилей. Клиент просматривает историю обслуживания своих автомобилей.

Должна быть возможность получения сводных отчётов на основе информации из БД: выручка по услугам за любой временной период, востребованность услуг детейлинг центра, перечень обслуженных автомобилей за любой выбранный период, перечень услуг, предоставленных клиенту по выбранному автомобилю за любой период.

Дополнительные требования к разрабатываемой системе:

- Авторизация пользователей (использование Microsoft Identity);

- Интерфейс и доступный функционал зависит от роли пользователя (3 роли);

- Адаптивный интерфейс;

- Валидация данных на стороне клиента (на предмет корректности данных) и на стороне сервера (на предмет соответствия данных логике работы);

- Формирование отчётов в виде файлов Excel таблиц и pdf-файлов;

- Отправка уведомлений пользователям в виде e-mail писем.

Таким образом, для разработки информационной системы детейлинг центра необходимо учесть все требования к базе данных и интерфейсу программы, а также выбрать подходящие инструменты для разработки.

1. Разработка моделей данных

2.1 Концептуальная модель данных

Концептуальная модель хранилища данных представляет собой описание главных (основных) сущностей и отношений между ними. Концептуальная модель является отражением предметных областей, в рамках которых планируется построение хранилища данных.

При проектировании концептуальной модели структурируют данные и выявляют взаимосвязи между ними, без рассмотрения особенностей реализации и вопросов эффективности обработки, поэтому концептуальная модель не является полностью подходящей для дальнейшей разработки, все таблицы должны быть нормализованы для реляционной базы данных. Составленная концептуальная модель представлена на рисунке 1 приложения 1.

2.2 Логическая модель данных

Логическая модель расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязи между ними.

Концептуальная модель изменяется так, чтобы она могла быть обеспечена конкретной моделью данных.

В результате формируется логическая модель. Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения.

Логическая модель может быть реляционной, иерархической или сетевой.

В качестве способа организации информационной базы выбрана реляционная база данных. Именно такой способ хранения всех данных является наиболее подходящим для проектируемой информационной системы по следующим причинам:

− наглядность модели для пользователя: все данные в реляционной модели представлены в табличной форме;

− независимость данных от программного продукта для их обработки;

− реляционные базы данных являются наиболее распространенными среди разработчиков ПО, следовательно, использование этих баз позволит сэкономить время и бюджет на внедрение нового типа БД.

Составленная логическая модель представлена на рисунке 2 приложения 1.

* 1. Физическая модель данных

Физические модели данных служат для отображения моделей данных. Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово "логический" введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных. Основными понятиями физической модели данных, используемыми для представления логической модели данных, являются поле, физическая запись, физический файл. В частности, логическая запись, состоящая из полей, может быть представлена в виде физической записи (из тех же полей), логический файл – в виде физического файла. Имена таблиц и колонок будут сгенерированы на основе сущностей и атрибутов логической модели, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые СУБД. Если в имени сущности или атрибута встречается пробел, он заменяется на символ «\_».

Физическая модель описывает способ хранения данных в базе данных. В физической модели мы учитываем типы данных, индексы, ограничения целостности и другие технические детали.

Составленная физическая модель представлена на рисунке 3 приложения 1.

В соответствии с данными моделями с помощью APS.NET Core MVC были сгенерированы следующие пять таблиц:

Первая таблица называется «Clients», в ней будет находиться информация об о клиентах детейлинг центра.

Таблица 1 – таблица «Clients».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ClientID | PRIMARY KEY, INTEGER, NOT NULL |
| Surname | NVARCHAR(max), NOT NULL |
| Name | NVARCHAR(max), NOT NULL |
| Patronym | NVARCHAR(max), NOT NULL |
| Phone | NVARCHAR(max), NOT NULL |

Вторая таблица называется «Automobiles», в ней будет находиться информация об автомобилях клиентов детейлинг центра.

Таблица 2 – таблица «Automobiles».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| AutomobileID | PRIMARY KEY, INTEGER, NOT NULL |
| Mark | NVARCHAR(max), NOT NULL |
| Model | NVARCHAR(max), NOT NULL |
| Gosnumber | NVARCHAR(max), NOT NULL |
| ClientId | FOREIGN KEY, INTEGER, NOT NULL |

Третья таблица называется «Services», в ней будет находиться информация об услугах детейлинг центра.

Таблица 3 – таблица «Services».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ServiceID | PRIMARY KEY, INTEGER, NOT NULL |
| Servicename | NVARCHAR(max), NOT NULL |
| Duration | INTEGER, NOT NULL |
| Price | INTEGER, NOT NULL |
| Description | NVARCHAR(max), NULL |

Четвёртая таблица называется «Employers», в ней будет находиться информация о сотрудниках детейлинг центра.

Таблица 4 – таблица «Employers».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| EmployerID | PRIMARY KEY, INTEGER, NOT NULL |
| Surname | NVARCHAR(max), NOT NULL |
| Name | NVARCHAR(max), NOT NULL |
| Patronym | NVARCHAR(max), NOT NULL |
| Phone | NVARCHAR(max), NOT NULL |
| City | NVARCHAR(max), NOT NULL |
| Street | NVARCHAR(max), NOT NULL |
| Housenumber | INTEGER, NOT NULL |
| Apartmentnumber | INTEGER, NULL |
| Photo | NVARCHAR(max), NULL |

Пятая таблица называется «ProvidedServices», в ней будет находиться информация об оказанных детейлинг центром услугах.

Таблица 5 – таблица «ProvidedServices».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ProvidedServicesID | PRIMARY KEY, INTEGER, NOT NULL |
| ServiceId | FOREIGN KEY, INTEGER, NULL |
| EmployerId | FOREIGN KEY, INTEGER, NULL |
| AutomobileId | FOREIGN KEY, INTEGER, NULL |
| dateTime | DATETIME2(7), NOT NULL |

1. Проектирование работы системы

В данном разделе содержится информация о проектировании работы “ИС детейлинг центра”, разработанной в текущей курсовой работе. В первую очередь было необходимо определить основные компоненты системы и их взаимодействие.

При проектировании работы системы детейлинг центра нужно уделить особое внимание архитектуре приложения и его функциональности. В соответствии с техническим заданием, система должна быть создана с использованием технологии ASP.NET Core MVC и базs данных SQL Server.

Первоначальным этапом проектирования является разработка моделей данных. Для учета клиентов, обслуживаемых автомобилей, оказываемых услуг, сотрудников и оказанных услуг необходимо создать соответствующие сущности и связи между ними. При создании моделей необходимо также предусматривать связи между ними, такие как связь "один ко многим" между клиентами и автомобилями.

Далее следует создание CRUD операций для администратора и сотрудника системы. Это позволит администратору управлять данными о сотрудниках и услугах, а сотруднику управлять данными о клиентах, их автомобилях и оказываемых услугах. В рамках создания CRUD функционала важно предусмотреть возможности добавления, просмотра, обновления и удаления записей.

Для клиента системы необходимо предусмотреть функционал регистрации в системе, просмотра оказанных услуг для автомобилей. Для обеспечения безопасности данных пользователей необходимо реализовать механизм аутентификации и авторизации.

Кроме того, система должна предоставлять возможность генерации сводных отчетов на основе данных из базы данных. Эти отчеты могут включать информацию о выручке по услугам за любой временной период, востребованность услуг детейлинг центра, перечень обслуженных автомобилей за любой выбранный период, перечень услуг, предоставленных клиенту по выбранному автомобилю за любой период.

Интерфейс приложения должен быть понятным и удобным для пользователей. Для этого рекомендуется использовать современные принципы дизайна пользовательского интерфейса и обеспечить его дружественность и удобство использования.

Система будет использовать базу данных для хранения всех необходимых данных. База данных будет содержать таблицы для учета автомобилей, клиентов, услуг, сотрудников и оказанных услуг.

Клиентская часть будет реализована с использованием технологии ASP.NET Core MVC. Она будет отвечать за отображение пользовательского интерфейса и взаимодействие с серверной частью. Серверная часть будет обрабатывать запросы от клиентской части, выполнять CRUD операции с базой данных и генерировать отчеты.

Проектирование работы информационной системы детейлинг центра включает в себя проектирование базы данных, разработку пользовательского интерфейса, создание CRUD операций для администратора, сотрудника. Реализацию функционала для пользователей и создание механизма генерации отчетов. Важно уделить внимание архитектуре приложения, безопасности данных и удобству использования для конечных пользователей.

1. Разработка и реализация системы

В данном разделе будет описан процесс разработки и реализации ИС детейлинг центра.

Каждая модель была разработана с учетом специфики данных и их взаимосвязей.

Всего было разработано пять моделей. Некоторые модели содержат внешние ключи, связывающие их с другими моделями. Для каждой из разработанных моделей был создан отдельный класс.

Модель «Клиент» представляет собой класс для хранения информации о клиенте детейлинг центра. О клиенте должна храниться следующая информация: ФИО и номер телефона. Нужно учитывать, что информация о клиенте будет использована и в модели для автомобиля, т.е. будет создан внешний ключ, связывающий автомобиль и клиента. Также необходимо проверять корректность ввода номера телефона. Для модели была продумана следующая структура класса:

- ClientID – Уникальный идентификатор клиента;

- Surname – Фамилия клиента;

- Patronym – Отчество клиента;

-Phone – Номер телефона клиента.

Реализация модели:

Листинг кода 1 – Класс для реализации модели Клиент Client.cs

public class Client

{

public int ClientID { get; set; }

[DisplayName("Фамилия")]

public string Surname { get; set; }

[DisplayName("Имя")]

public string Name { get; set; }

[DisplayName("Отчество")]

public string Patronym { get; set; }

[DisplayName("Номер телефона")]

[RegularExpression(@"^((8|\+7)[\- ]?)?(\(?\d{3}\)?[\- ]?)?[\d\- ]{7,10}$", ErrorMessage = "Неверно введён номер телефона. Необходимо ввести в следующем формате 8(888)888-88-88")]

public string Phone { get; set; }

}

Модель «Автомобиль» представляет собой класс для хранения информации об автомобиле клиента детейлинг центра. Об автомобиле необходима следующая информация: марка, модель и гос.номер. Как упоминалось выше, должна храниться информация о владельце автомобиля (клиенте детейлинг центра), в связи с чем необходимо связать эти модели с помощью поля класса и создания внешнего ключа, связывающего автомобиль и клиента. Нужно учитывать, что информация об автомобиле будет использована и в модели для оказанной услуги, т.е. будет создан внешний ключ, связывающий автомобиль и оказанную услугу. Также необходимо проверять корректность ввода гос. номера. Для модели была продумана следующая структура класса:

- AutomobileID – Уникальный идентификатор автомобиля;

- Mark – Марка автомобиля;

- Model – Модель автомобиля;

- Gosnumber – Гос.номер автомобиля;

- ClientId – Идентификатор владельца (клиента);

- Client – экземпляр класса Client, соответствующий идентификатору клиента (ClientId).

Реализация модели:

Листинг кода 2 – Класс для реализации модели Автомобиль Automobile.cs:

public class Automobile

{

public int AutomobileID { get; set; }

[DisplayName("Марка")]

public string Mark { get; set; }

[DisplayName("Модель")]

public string Model { get; set; }

[DisplayName("Гос. номер")]

[RegularExpression(@"^[ABEKMHOPCTYX]\d{3}(?<!000)[ABEKMHOPCTYX]{2}\d{1,3}rus$", ErrorMessage = "Неверно введён гос. номер. Необходимо ввести в следующем формате A111AA222rus")]

public string Gosnumber { get; set; }

public int? ClientId { get; set; }

public Client? Client { get; set; }

}

Модель «Сотрудник» представляет собой класс для хранения информации о сотруднике детейлинг центра. О сотруднике должна храниться следующая информация: ФИО, номер телефона, адрес и фотография. Нужно учитывать, что информация о сотруднике будет использована и в модели для оказанной услуги, т.е. будет создан внешний ключ, связывающий сотрудника и оказанную услугу. Также необходимо проверять корректность ввода номера телефона. Для модели была продумана следующая структура класса:

- EmployerID – Уникальный идентификатор сотрудника;

- Surname – Фамилия сотрудника;

- Patronym – Отчество сотрудника;

- Phone – Номер телефона сотрудника;

- City – Город проживания сотрудника;

- Street – Улица проживания сотрудника;

- Housenumber – Номер дома проживания сотрудника;

- Appartmentnumber – Номер квартиры проживания сотрудника;

- Photo – Фотография сотрудника.

Реализация модели:

Листинг кода 3 – Класс для реализации модели Сотрудник Employer.cs:

public class Employer

{

public int EmployerID { get; set; }

[DisplayName("Фамилия")]

public string Surname { get; set; }

[DisplayName("Имя")]

public string Name { get; set; }

[DisplayName("Отчество")]

public string Patronym { get; set; }

[DisplayName("Номер телефона")]

[RegularExpression(@"^((8|\+7)[\- ]?)?(\(?\d{3}\)?[\- ]?)?[\d\- ]{7,10}$", ErrorMessage = "Неверно введён номер телефона. Необходимо ввести в следующем формате 8(888)888-88-88")]

public string Phone { get; set; }

[DisplayName("Город")]

public string City{ get; set; }

[DisplayName("Улица")]

public string Street { get; set; }

[DisplayName("Номер дома")]

public int Housenumber { get; set; }

[DisplayName("Номер квартиры")]

public int? Appartmentnumber { get; set; }

public string? Photo { get; set; }

}

Модель «Услуга» представляет собой класс для хранения информации об услуге детейлинг центра. Об услуге должна храниться следующая информация: название, длительность, стоимость и описание. Нужно учитывать, что информация об услуге будет использована и в модели для оказанной услуги, т.е. будет создан внешний ключ, связывающий услугу и оказанную услугу. Для модели была продумана следующая структура класса:

- ServiceID – Уникальный идентификатор услуги;

- Servicename – Название услуги;

- Duration – Длительность услуги;

- Price – Стоимость услуги;

- Description – Описание услуги.

Реализация модели:

Листинг кода 4 – Класс для реализации модели Услуга Service.cs:

public class Service

{

public int ServiceID { get; set; }

[DisplayName("Название")]

public string Servicename { get; set; }

[DisplayName("Длительность")]

public int Duration { get; set; }

[DisplayName("Стоимость")]

public int Price { get; set; }

[DisplayName("Описание")]

public string? Description { get; set; }

}

Модель «Оказанная услуга» представляет собой класс для хранения информации об оказанной услуге детейлинг центра. Об оказанной услуге должна храниться дата оказания услуги.номер. Как упоминалось выше, должна храниться информация об автомобиле клиента, услуге и сотруднике, оказывавшем услугу, в связи с чем необходимо связать эти модели с помощью соответствующих полей классов и создания внешних ключей, связывающих оказанную услугу с услугой, автомобилем и сотрудником соответственно. Для модели была продумана следующая структура класса:

- ProvidedServiceID – Уникальный идентификатор оказанной услуги;

- ServiceId – Идентификатор услуги;

- Service – экземпляр класса Service, соответствующий идентификатору услуги (ServiceId);

- EmployerId – Идентификатор сотрудника;

- Employer – экземпляр класса Employer, соответствующий идентификатору сотрудника (EmployerId);

- AutomobileId – Идентификатор автомобиля;

- Automobile – экземпляр класса Automobile, соответствующий идентификатору автомобиля (AutomobileId);

- dateTime – Дата и время оказания услуги.

Реализация модели:

Листинг кода 5 – Класс для реализации модели Оказанная услуга ProvidedService.cs:

public class ProvidedService

{

public int ProvidedServiceID { get; set; }

public int? ServiceId { get; set; }

public int? EmployerId { get; set; }

public int? AutomobileId { get; set; }

[DisplayName("Дата и время оказания услуги")]

public DateTime dateTime { get; set; }

public Service? Service { get; set; }

public Employer? Employer { get; set; }

public Automobile? Automobile { get; set; }

}

С помощью стандартных средств фреймворка сгенерированы контроллеры с реализованными CRUD - операциями для работы с БД и web-страницы, связанные с контроллерами, для удобной работы пользователя.

Одна из целей курсовой работы - создание функционала регистрации и авторизации пользователей. Это является важной частью любой системы, обеспечивающей безопасность данных и управление доступом. В качестве основы для реализации данной функции использовался механизм ASP.NET Core Identity.

Настройка модели пользователя: ASP.NET Core Identity предоставляет базовый класс IdentityUser, который включает в себя стандартные поля, такие как UserName, Email, PhoneNumber и PasswordHash.

Реализация добавления свойств о пользователе:

С помощью стандартных средств фреймворка были сгенерированы БД для хранения данных о пользователях и web-страницы для регистрации и авторизации. Далее были внесены изменения в класс реализации модели пользователя для добавления дополнительных свойств: фамилии, имени и отчества. Также были перенастроена web-страница регистрации.

Листинг кода 6 – Класс для реализации модели пользователя:

public class DetApp3User : IdentityUser

{

public string Surname { get; set; }

public string Name { get; set; }

public string Patronym { get; set; }

}

Листинг кода 7 – Изменения в файле Register.cshtml:

<div class="form-floating mb-3">

<input asp-for="Input.Surname" class="form-control" autocomplete="username"

aria-required="true" placeholder="Surname" />

<label asp-for="Input.Surname">Surname</label>

<span asp-validation-for="Input.Surname" class="text-danger"></span>

</div>

<div class="form-floating mb-3">

<input asp-for="Input.Name" class="form-control" autocomplete="username"

aria-required="true" placeholder="Name" />

<label asp-for="Input.Name">Name</label>

<span asp-validation-for="Input.Name" class="text-danger"></span>

</div>

<div class="form-floating mb-3">

<input asp-for="Input.Patronym" class="form-control" autocomplete="username"

aria-required="true" placeholder="Patronym" />

<label asp-for="Input.Patronym">Patronym</label>

<span asp-validation-for="Input.Patronym" class="text-danger"></span>

</div>

После настройки механизма авторизации и аутентификации, информация о текущем пользователе может быть использована в контроллерах и представлениях для управления доступом и отображения соответствующих данных.

Внедрение системы управления учётными записями пользователей с использованием ASP.NET Core Identity значительно повышает безопасность и управляемость веб-приложения детейлинг центра. Настройка и интеграция Identity позволили создать гибкий и расширяемый механизм аутентификации и авторизации, обеспечивающий удобство и безопасность для пользователей.

Внедрение механизма создания отчетов также являлось одной из задач курсовой работы. Эта задача была решена с помощью использования сторонней библиотеки EPPlus для работы с файлами Excel. Эта библиотека позволяет легко создавать и модифицировать файлы Excel, что значительно упрощает процесс генерации отчетов.

При разработке и реализации системы были реализованы: регистрация и авторизация пользователей, генерация отчетов в формате Excel. Использование ASP.NET Core Identity обеспечило надежную систему управления пользователями, а библиотека EPPlus предоставила мощные инструменты для работы с файлами Excel.

1. Тестирование системы

Целью проведения тестирования является подтверждение реализации требуемой функциональной системы. Случаем, когда тестирование прошло успешно является совпадение с ожидаемым результатом.

В случае, когда приложение требует подтверждения действия пользователя появляется окно с кнопками подтверждения/отмены и уведомлением (информации, на основе которой пользователь должен сделать соответствующий выбор)

При обнаружении ошибки приложением, оно выводит сообщение, содержащее текст ошибки, на экран.

Тестирование было проведено для каждой из разработанных моделей, результаты проверки правильности работы приложения представлены в таблице 6. Демонстрация тестирования отражена в приложении 3.

Таблица 6 – Результаты тестирования разработанного приложения.

|  |  |  |
| --- | --- | --- |
| Выполняемое действие | Ожидаемый результат | Полученный результат |
| Запуск приложения | Отображение «домашней» страницы приложения. | Появление «домашней» страницы приложения. |
| Добавление записей. | Добавление записи в таблицу и отображение ее на странице | Записи были успешно добавлены в базу данных и отражены на странице. |
| Переходы между страницами | Корректные переходы между страницами. | Успешные переходы между страницами. |
| Нажатие кнопки «Создать отчёт» | Создание отчета и вывод диалогового окна для выбора места сохранения файла | Успешное создание отчета и сохранение его на устройстве пользователя |
| Нажатие кнопки  «Авторизация» | Открытие страницы авторизации с полями для заполнения | Открытие страницы авторизации с полями для заполнения |
| Авторизация пользователя | Авторизация пользователя и вход в профиль | Успешная авторизация пользователя вход в профиль |
| Нажатие кнопки  «Добавить» | Открытие страницы добавления с требуемыми полями | Открытие страницы с требуемыми полями |
| Нажатие кнопки  «Удалить» | Удаление записи из базы данных и из списка записей на странице | Успешное удаление записи из базы данных и с веб страницы |
| Нажатие кнопки «Подробнее» | Переход на страницу с данными о записи | Успешный переход на страницу с данными о записи |
| Нажатие кнопки  «Изменить» | Переход на страницу с заполненным полями о записи | Успешный переход на страницу с заполненным полями о записи |
| Добавление фотографии | Открытие диалогового окна для выбора фото, загрузка фото в базу и вывод на форму. | Успешное открытие диалогового окна для выбора фото, загрузка фото в базу и вывод на форму |

Результаты, полученные в ходе тестирования разработанного приложения, позволяют сделать заключение в том, что разработанная программа соответствует требованиям технического задания.

Заключение

В ходе выполнения курсового проекта была разработана распределенная информационная система для автоматизации деятельности детейлинг центра с использованием технологии ASP.NET Core MVC, созданы концептуальная, логическая и физическая модели данных. В результате работы создано приложение для ведения учета сотрудников детейлинг центра, оказываемых услуг, клиентов и их автомобилей, хранения информации об оказанных автомобилям услуг с возможностью получения отчетов на основе информации из БД. В приложении реализованы хранение изображений, валидация данных и генерация отчётов в соответствии с тематикой работы.

Подводя итоги, можно считать, что разработанное приложение соответствует требованиям технического задания.

Список литературы

1) Столбовский, Д. Н. Разработка Web-приложений ASP.NET с использованием Visual Studio .NET : учебное пособие / Д. Н. Столбовский. - 3-е изд. - Москва, Саратов.

2) Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. - 375 с.

3) Фримен Адам. ASP. NET MVC 5 с примерами на С# 5.0 для профессионалов : Вильямс, 2018, 736с.

4) Brian L Gorman. Practical Entity Framework: Database Access for Enterprise Applications. - Apress, 2020 - 433pp.

5) Эспозито Дино Разработка современных веб-приложений. Анализ предметных областей и технологий. — Вильямс, 2017, 464с.

Приложение 1

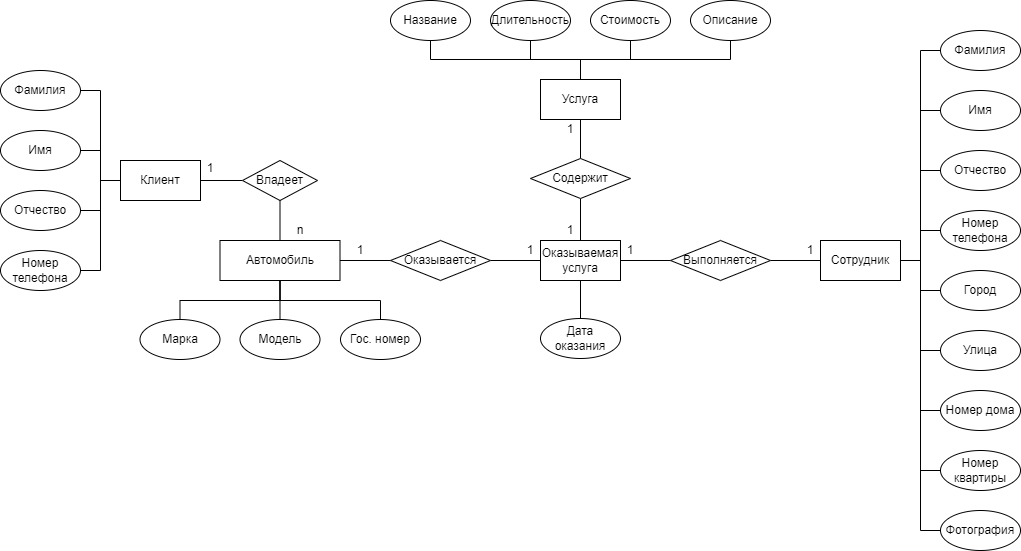


Рисунок 1 – Концептуальная модель данных

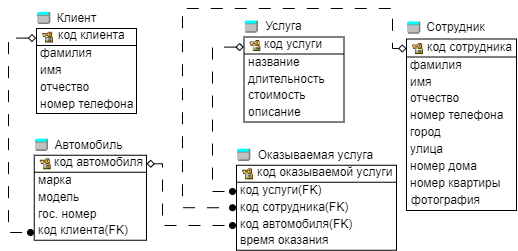


Рисунок 2 – Логическая модель данных

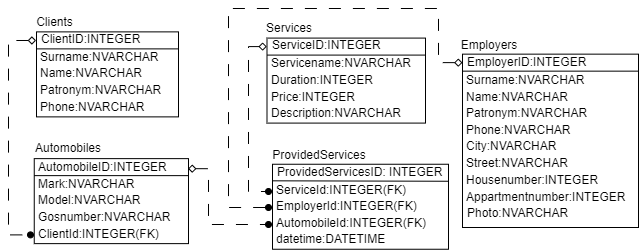


Рисунок 3 – Физическая модель данных

Приложение 2

Для подробного ознакомления с данным приложением можно использовать ссылку на репозиторий данного проекта: https:// <https://github.com/Mejioy/DetApp3>

Приложение 3

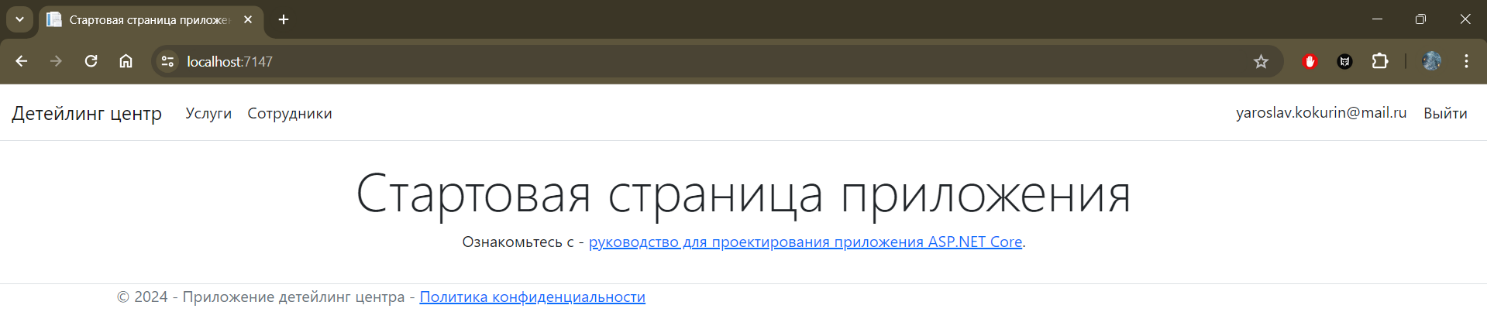


Рисунок 1 – Стартовая страница приложения

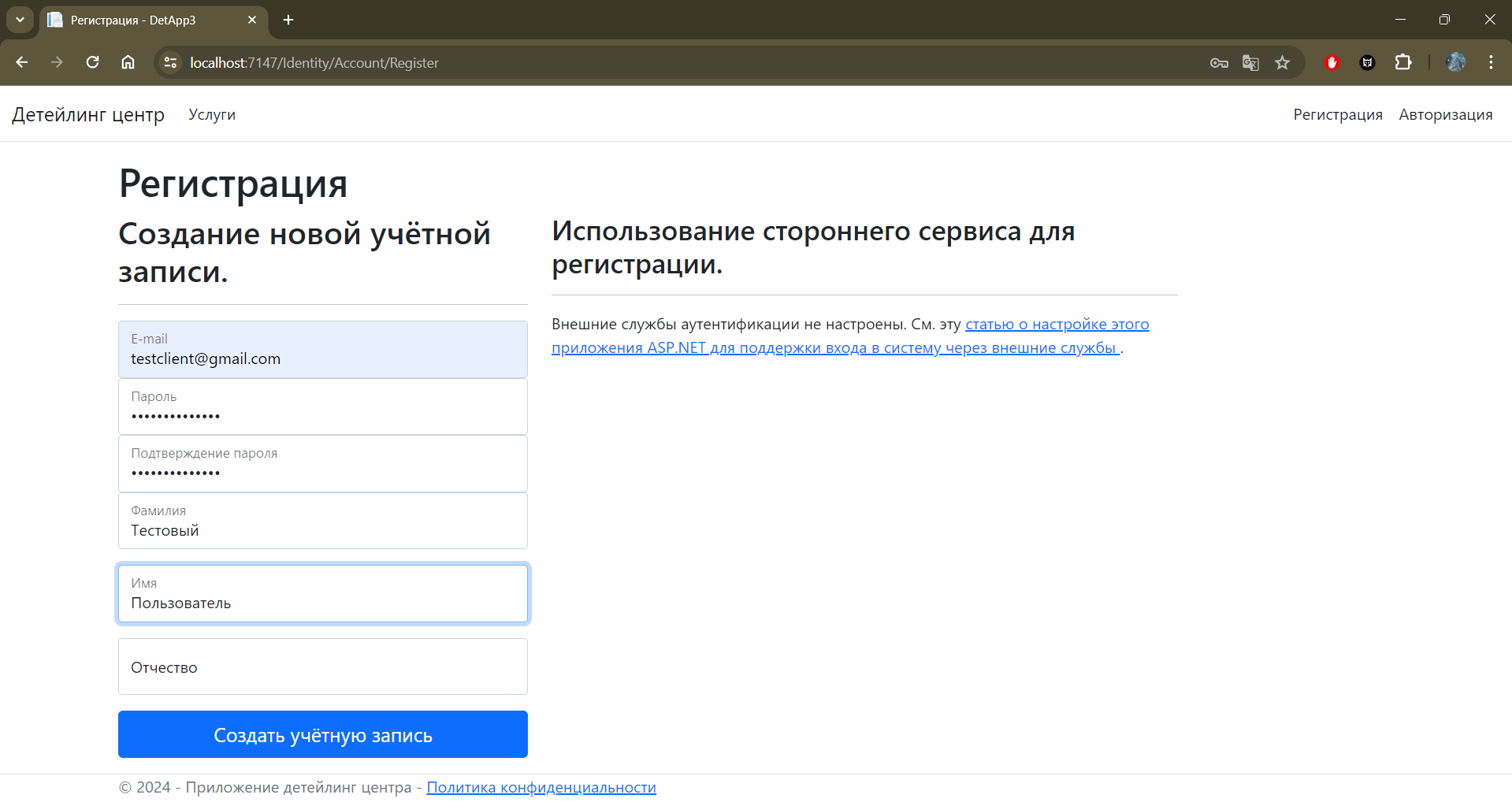


Рисунок 2 – Страница регистрации

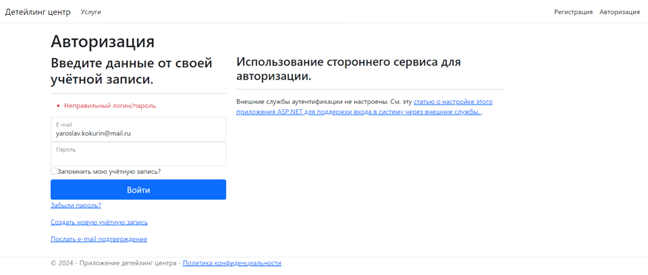


Рисунок 3 – Страница авторизации

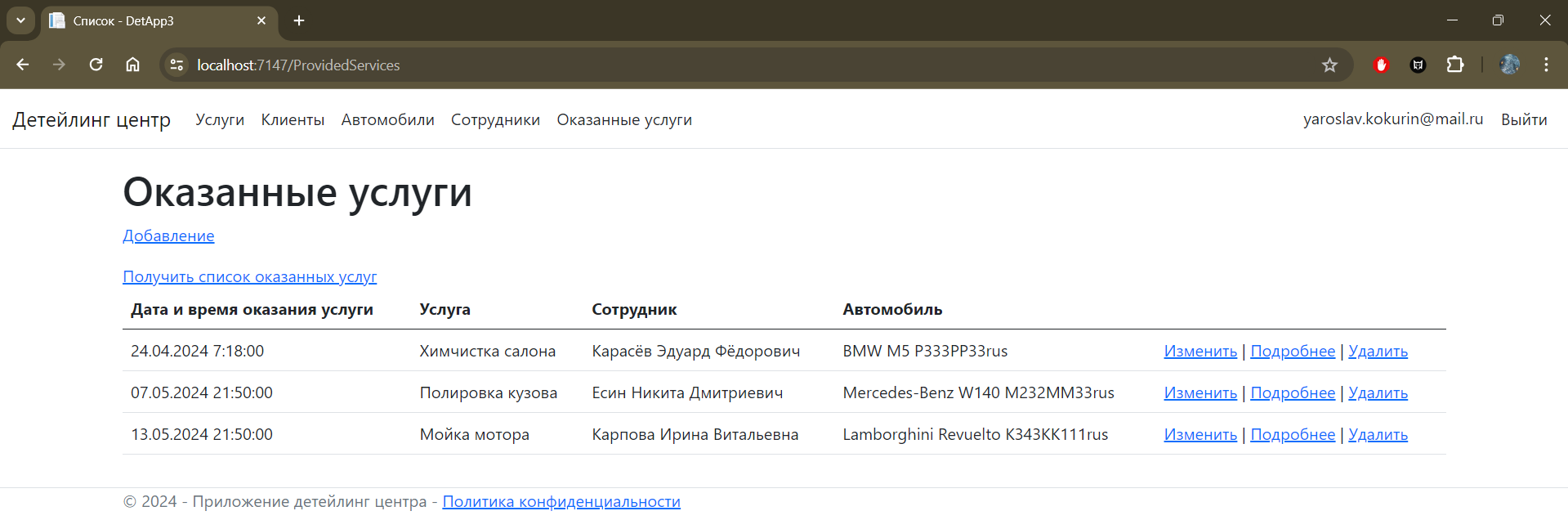


Рисунок 4 – Список оказанных услуг

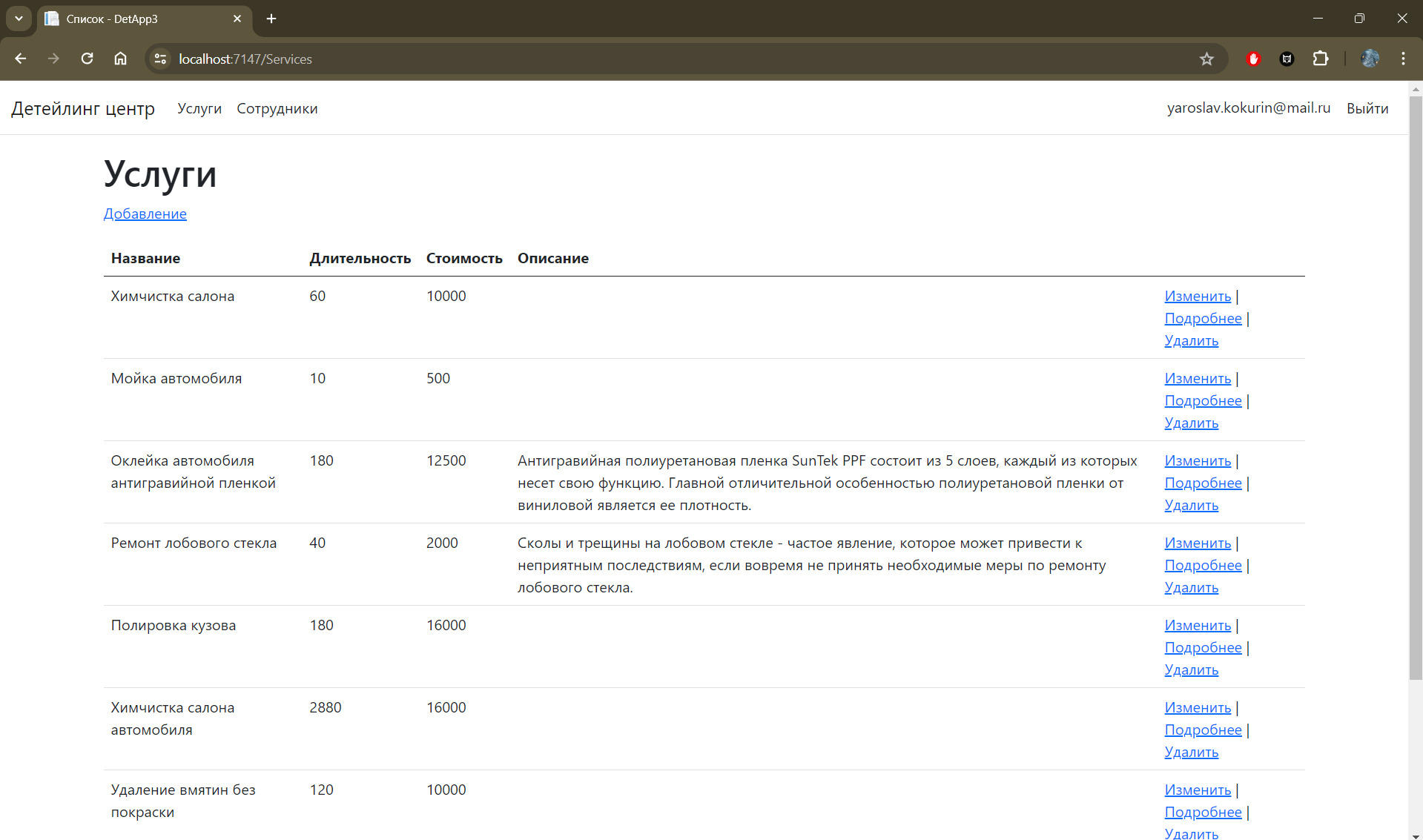


Рисунок 5 – Список оказываемых услуг

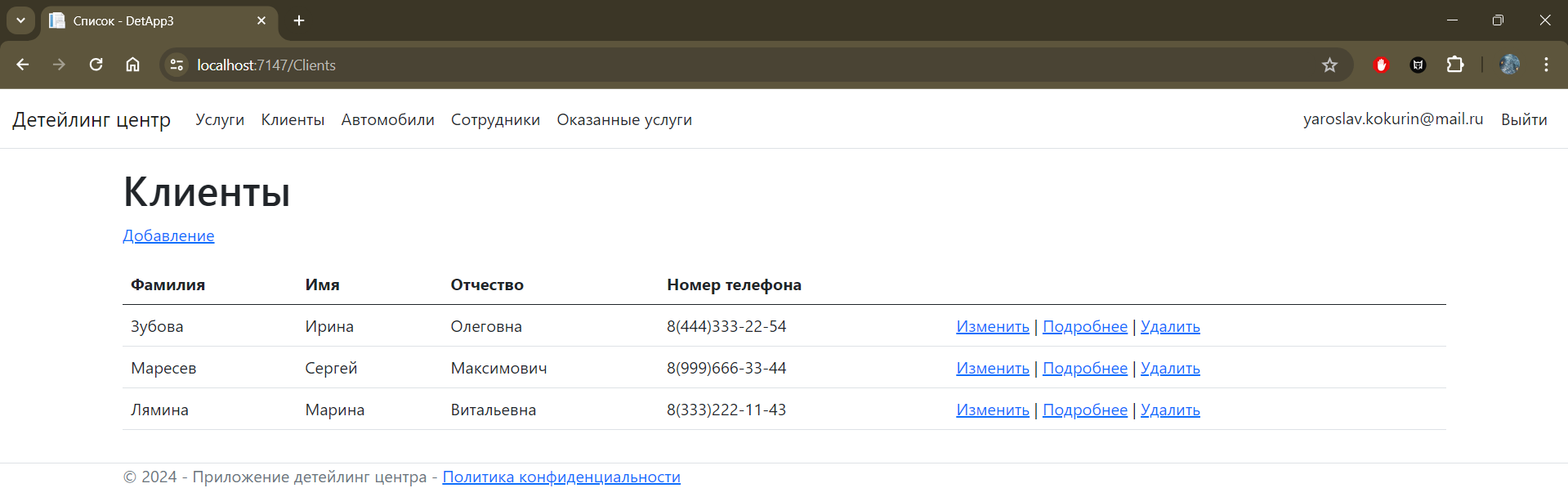


Рисунок 6 – Список клиентов

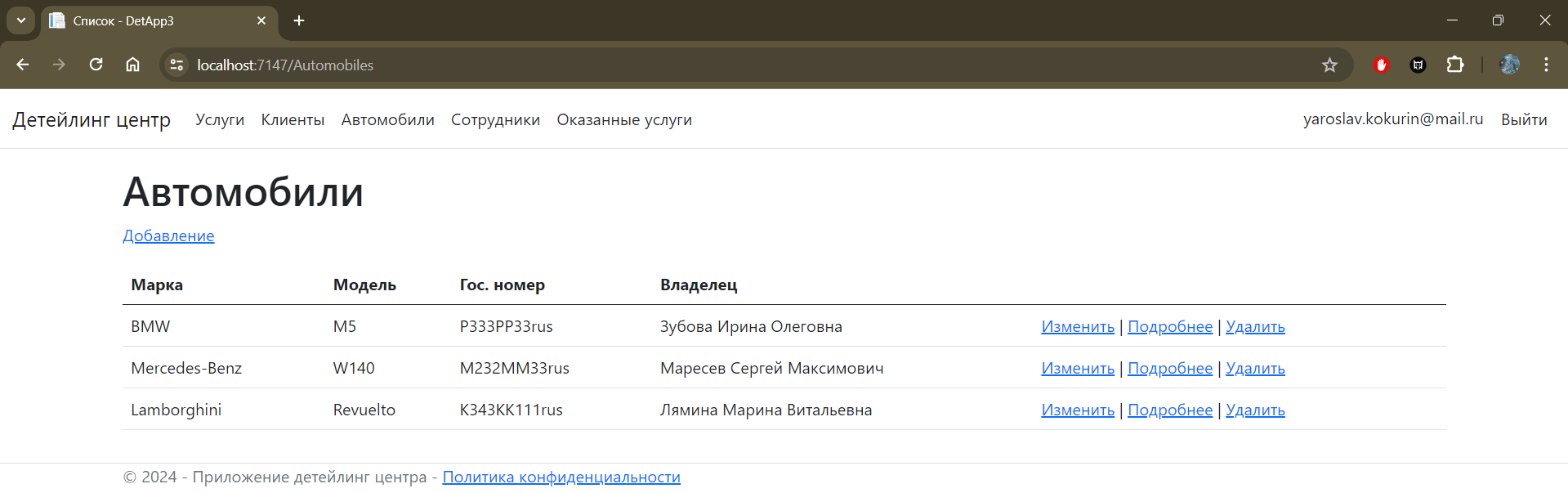


Рисунок 7 – Список автомобилей клиентов

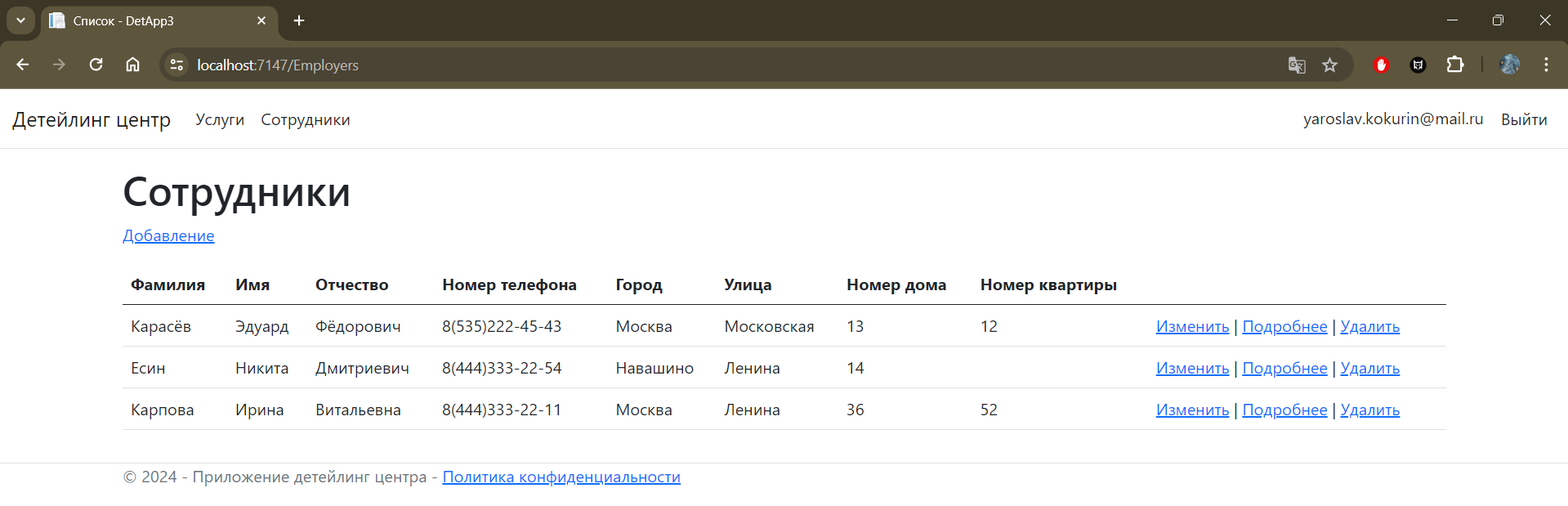


Рисунок 8 – Список сотрудников

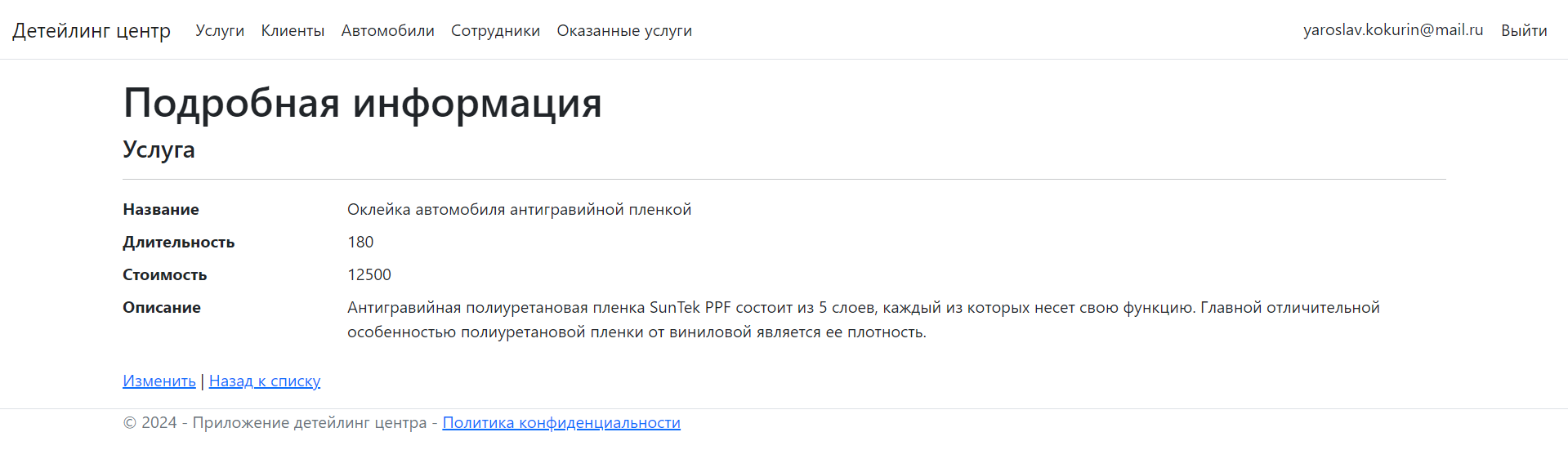


Рисунок 9 – Страница подробной информации об услуге

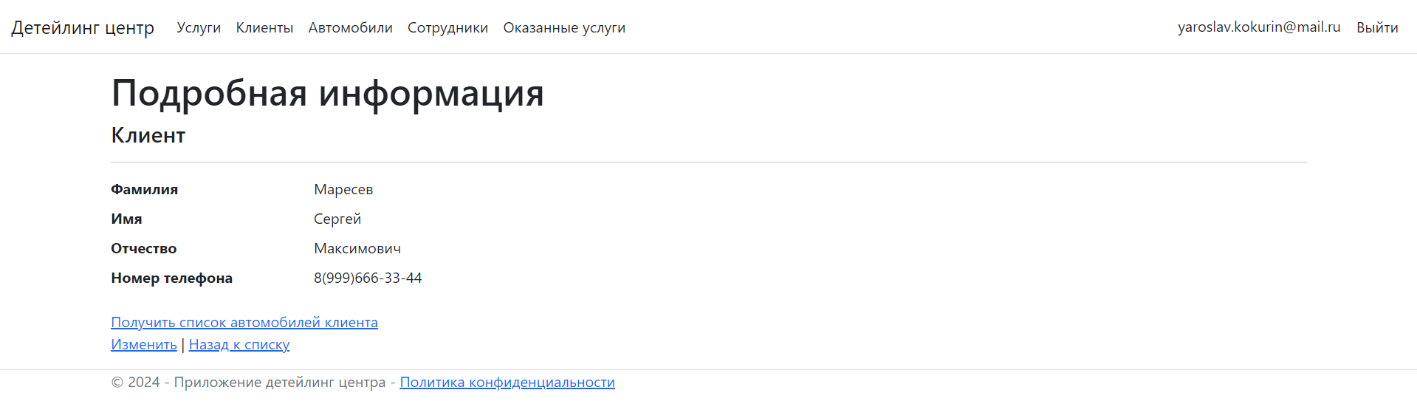


Рисунок 10 – Страница подробной информации о клиенте

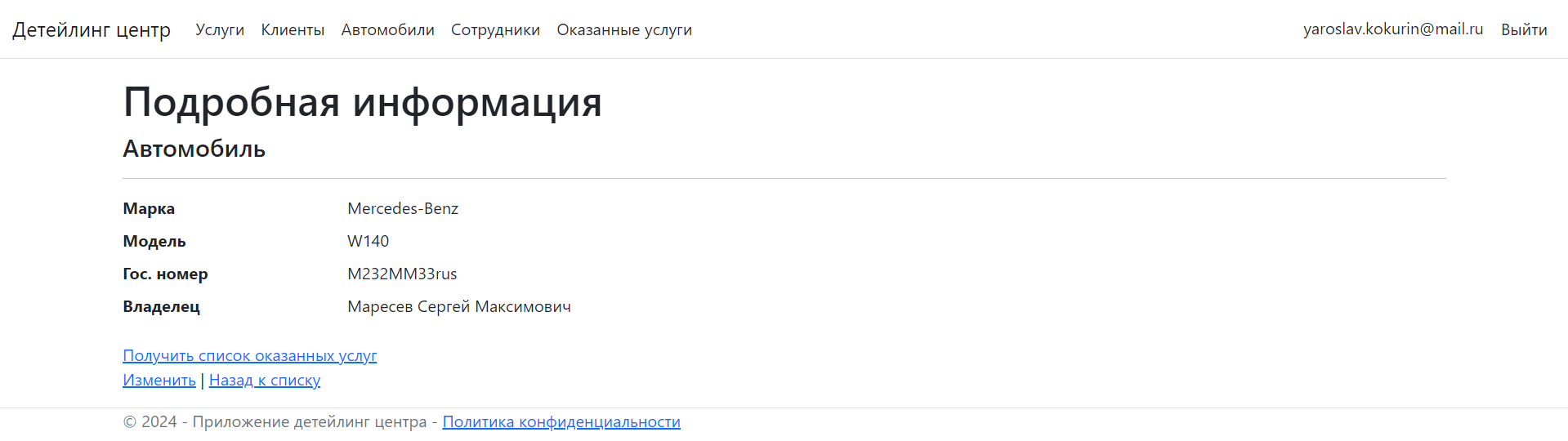


Рисунок 11 – Страница подробной информации об автомобиле

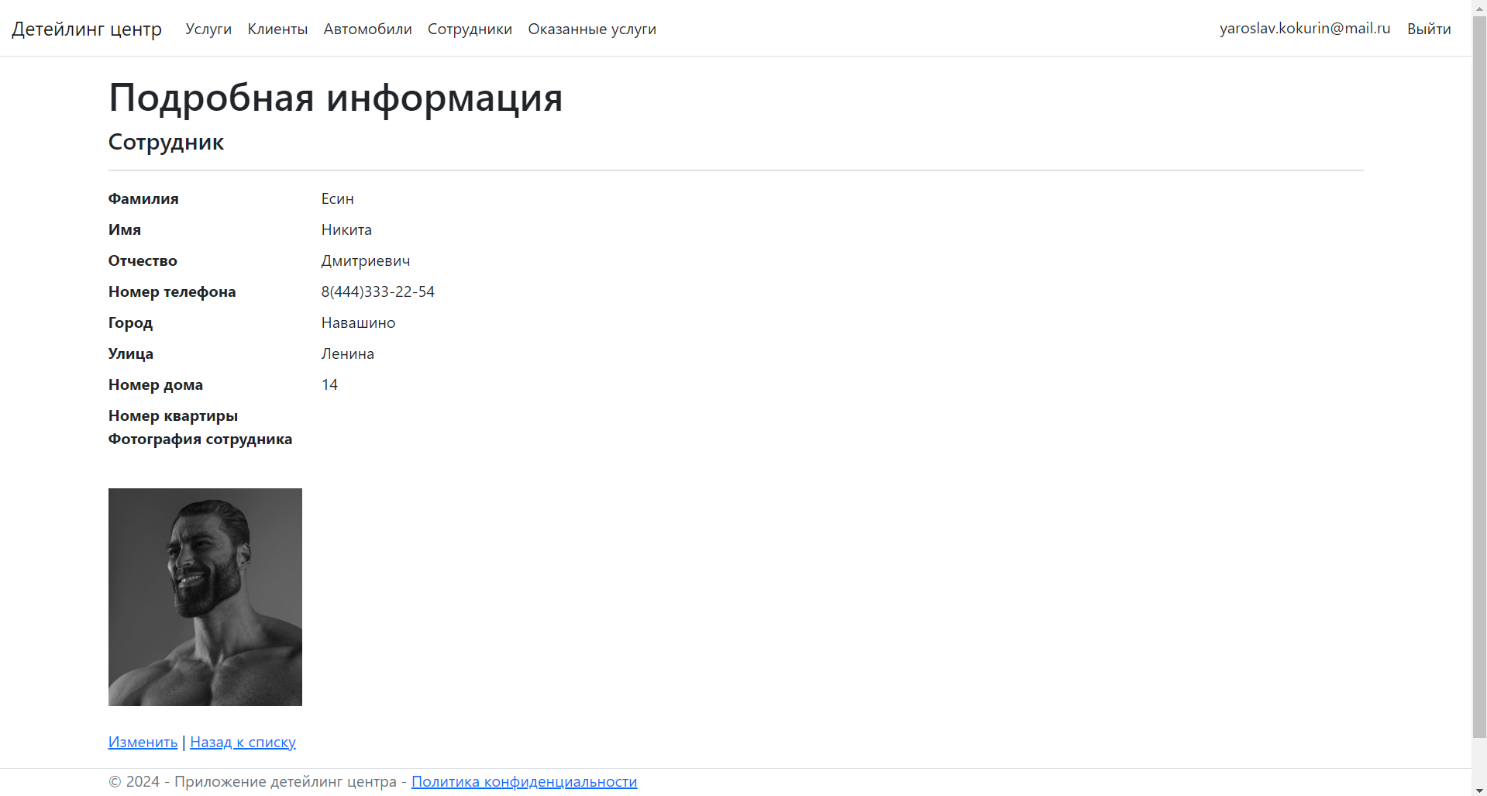


Рисунок 12 – Страница подробной информации о сотруднике

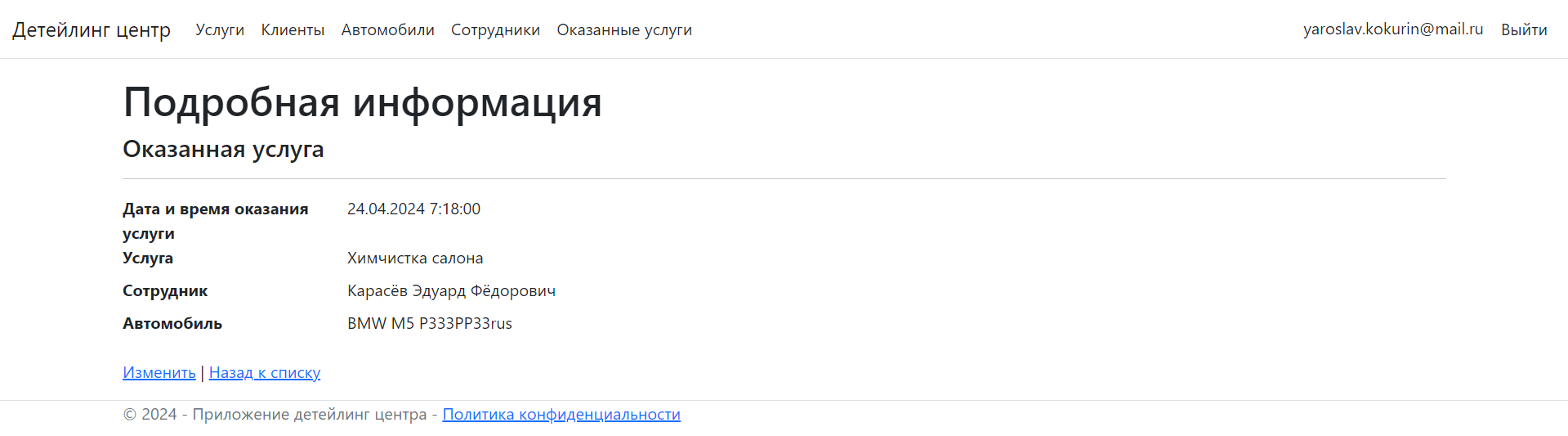


Рисунок 13 – Страница подробной информации об оказанной услуге

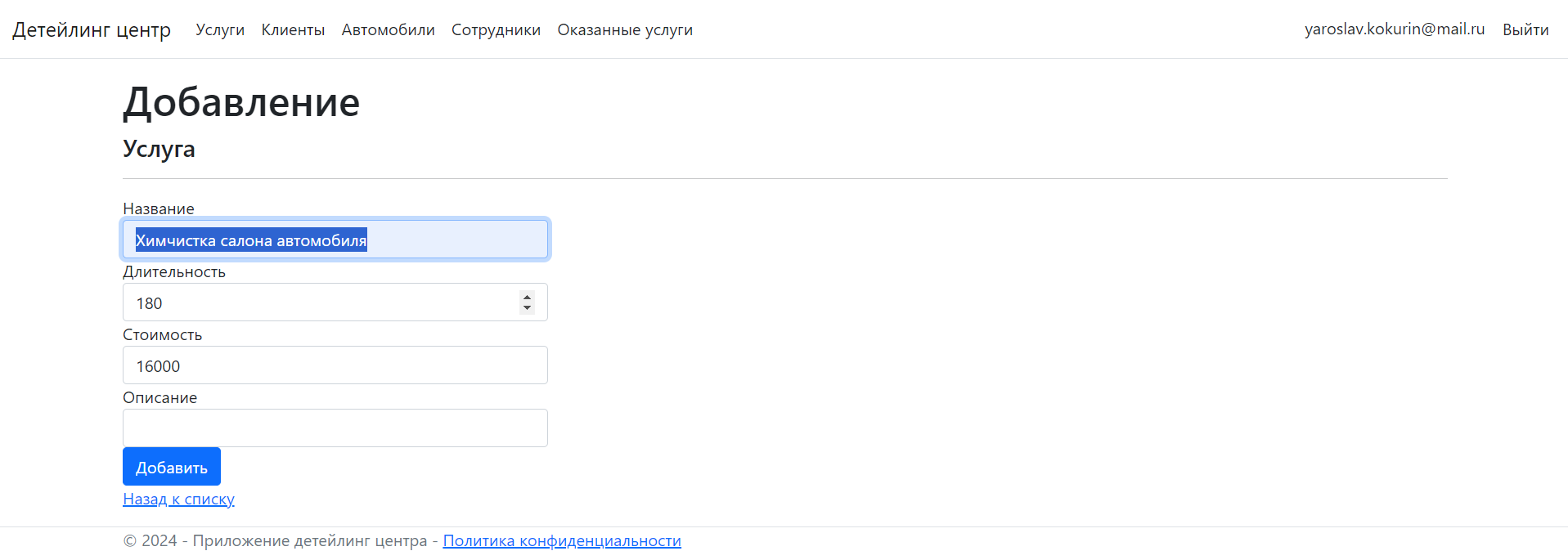


Рисунок 14 – Страница добавления услуги

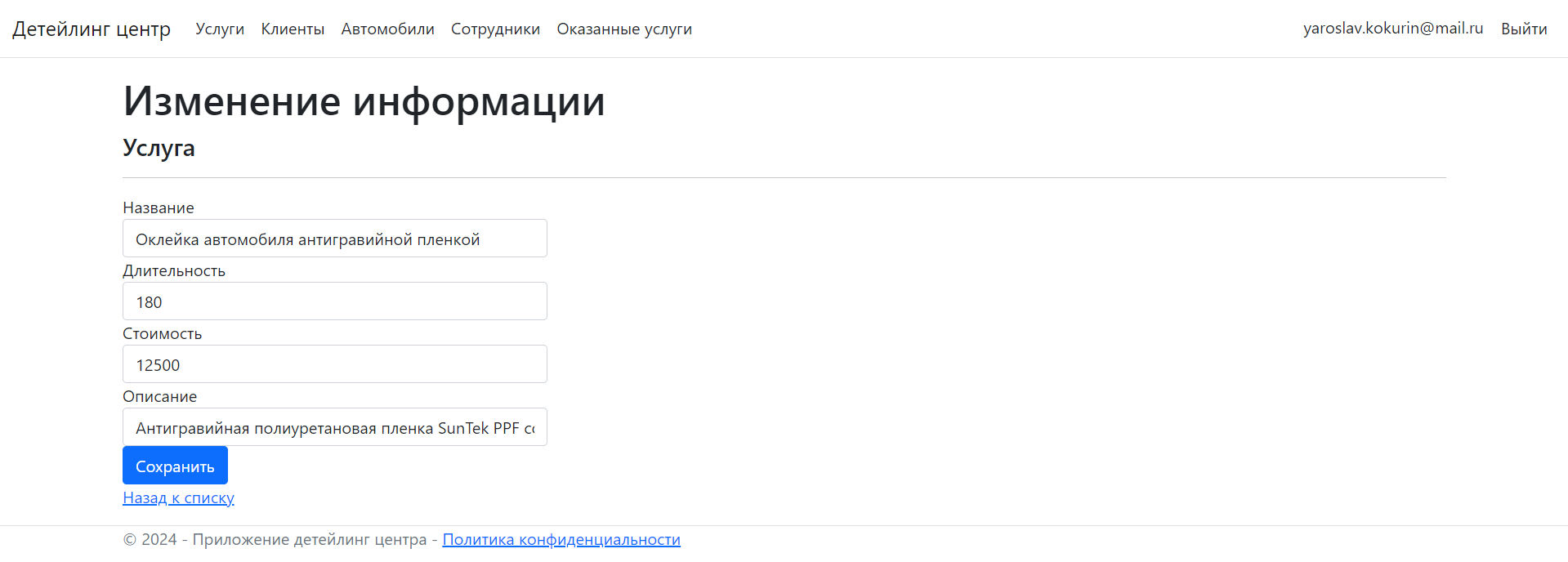


Рисунок 15 – Страница изменения информации об услуге

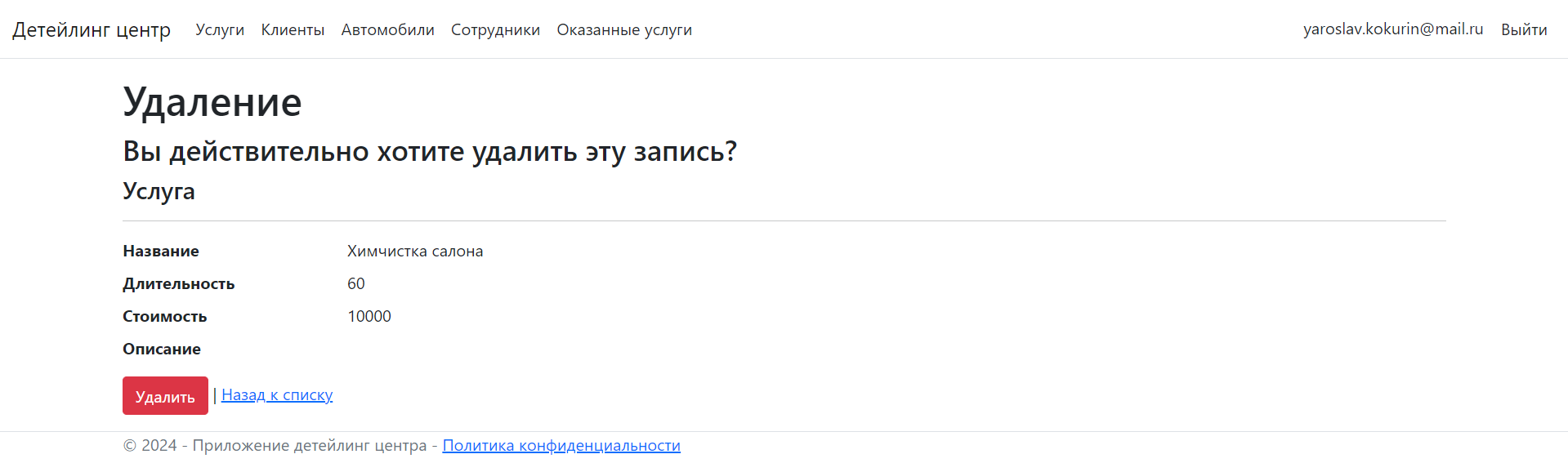


Рисунок 16 – Страница удаления услуги

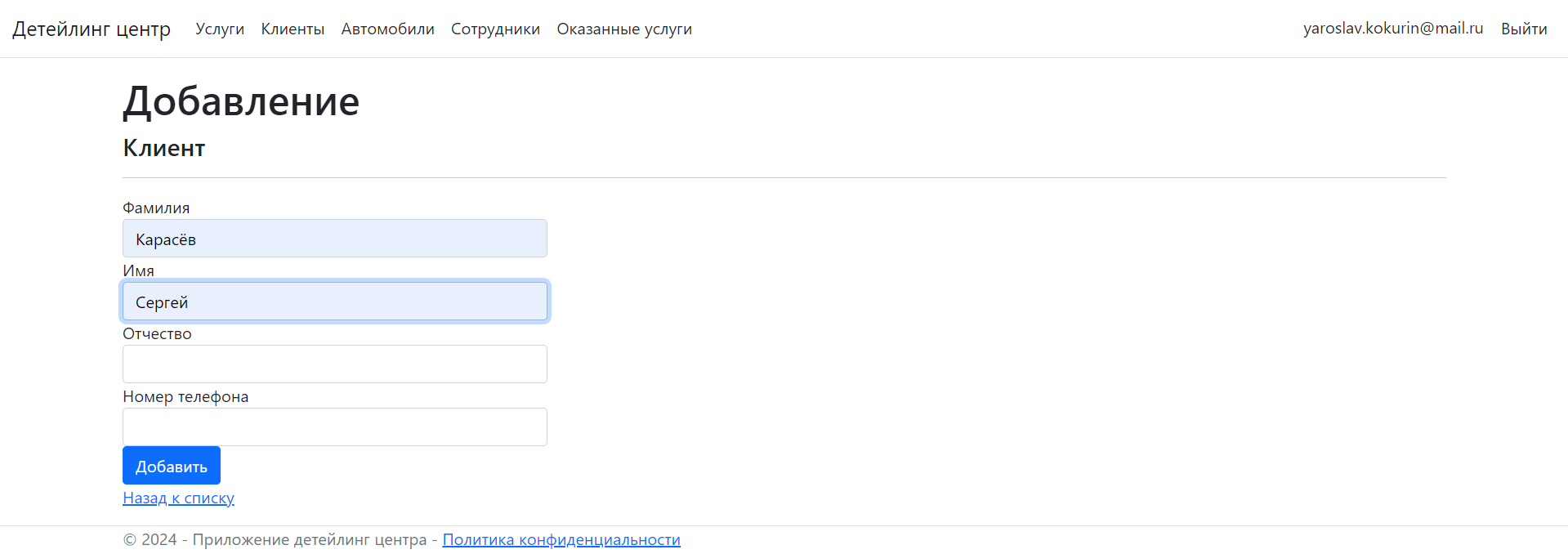


Рисунок 17 – Страница добавления клиента

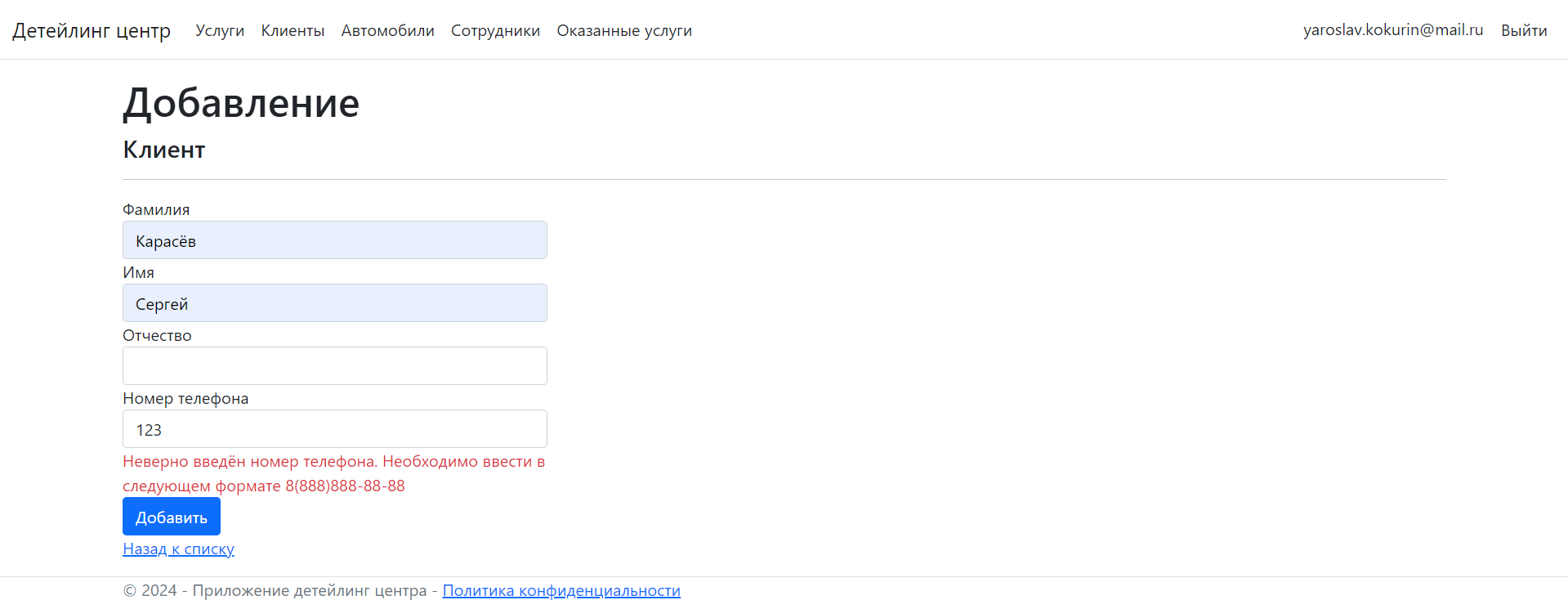


Рисунок 18 – Проверка ввода номера телефона на странице добавления клиента

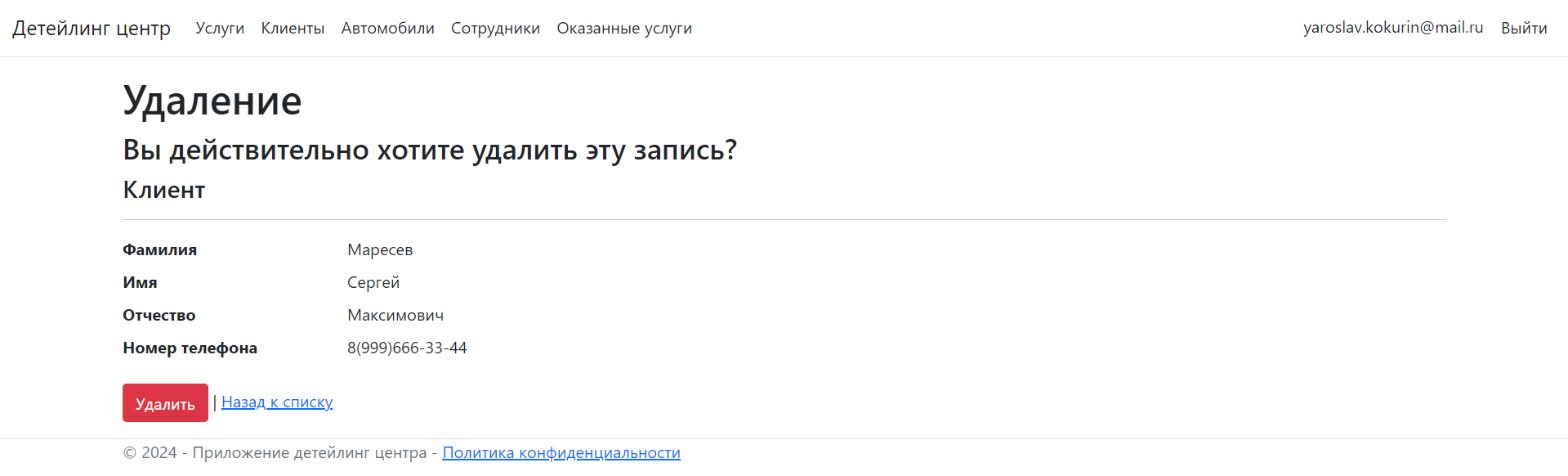


Рисунок 19 – Страница удаления клиента

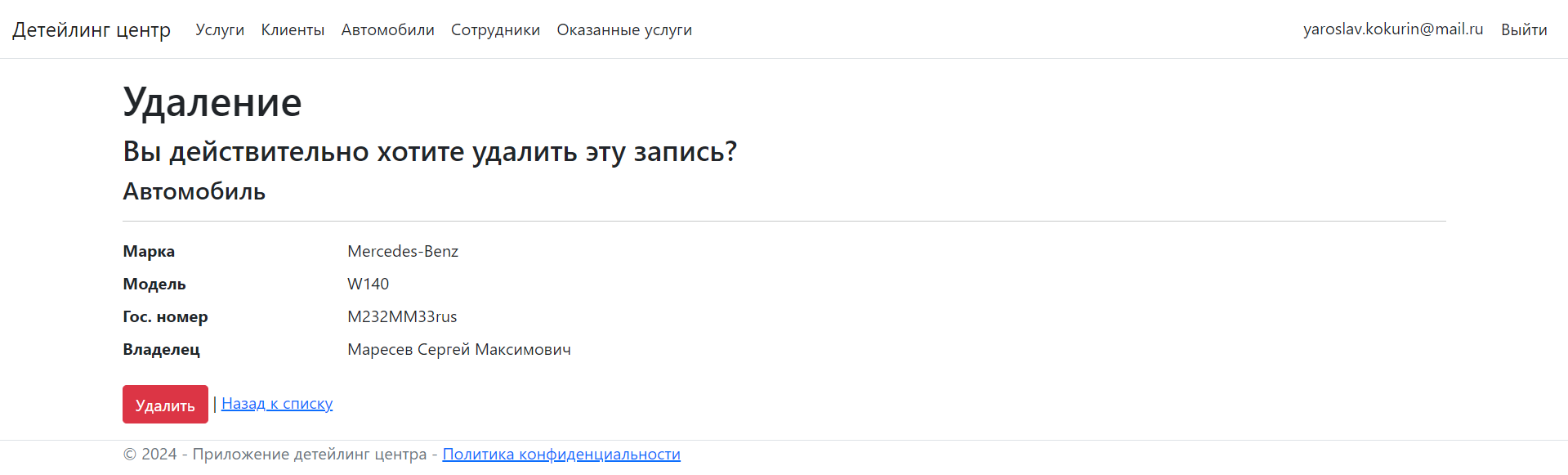


Рисунок 20 – Страница удаления автомобиля

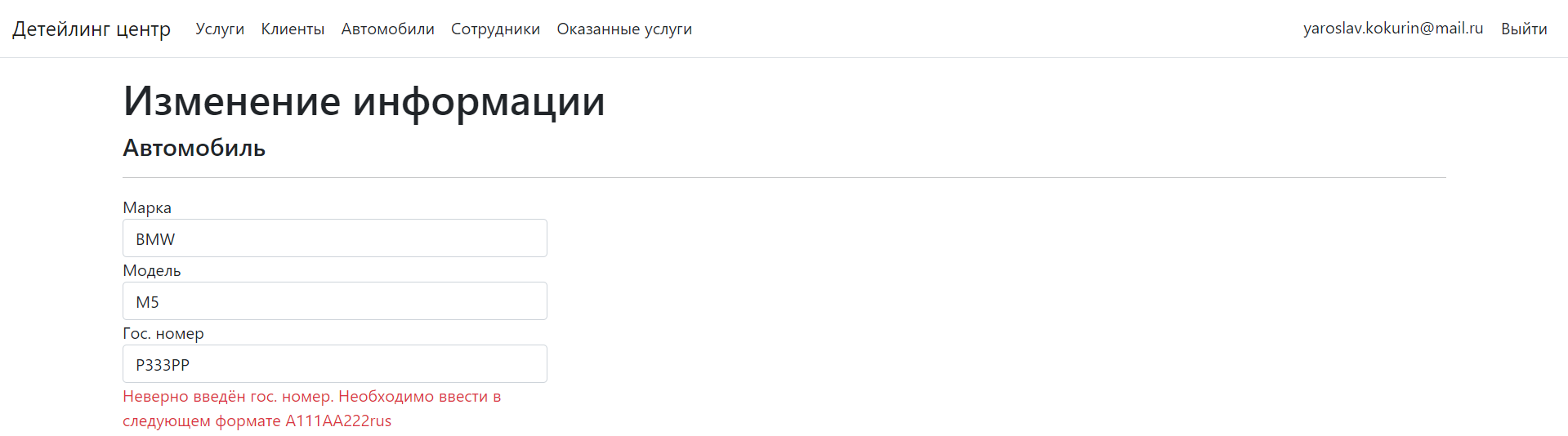


Рисунок 21 – Проверка ввода гос.номера на странице добавления автомобиля

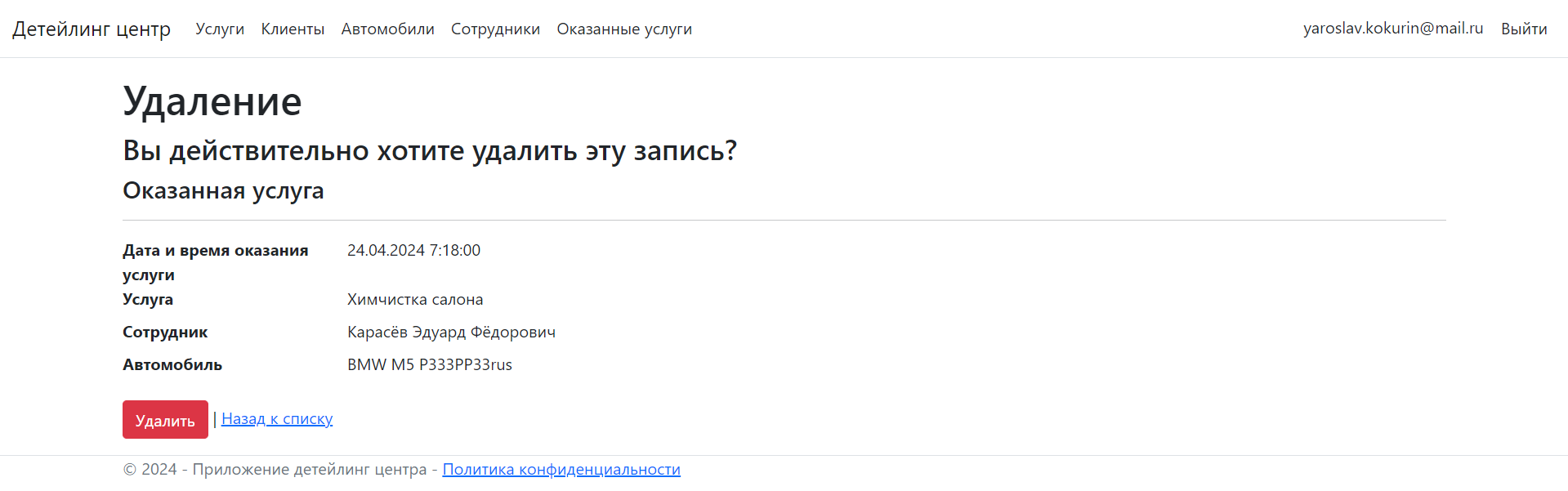


Рисунок 22 – Страница удаления оказанной услуги

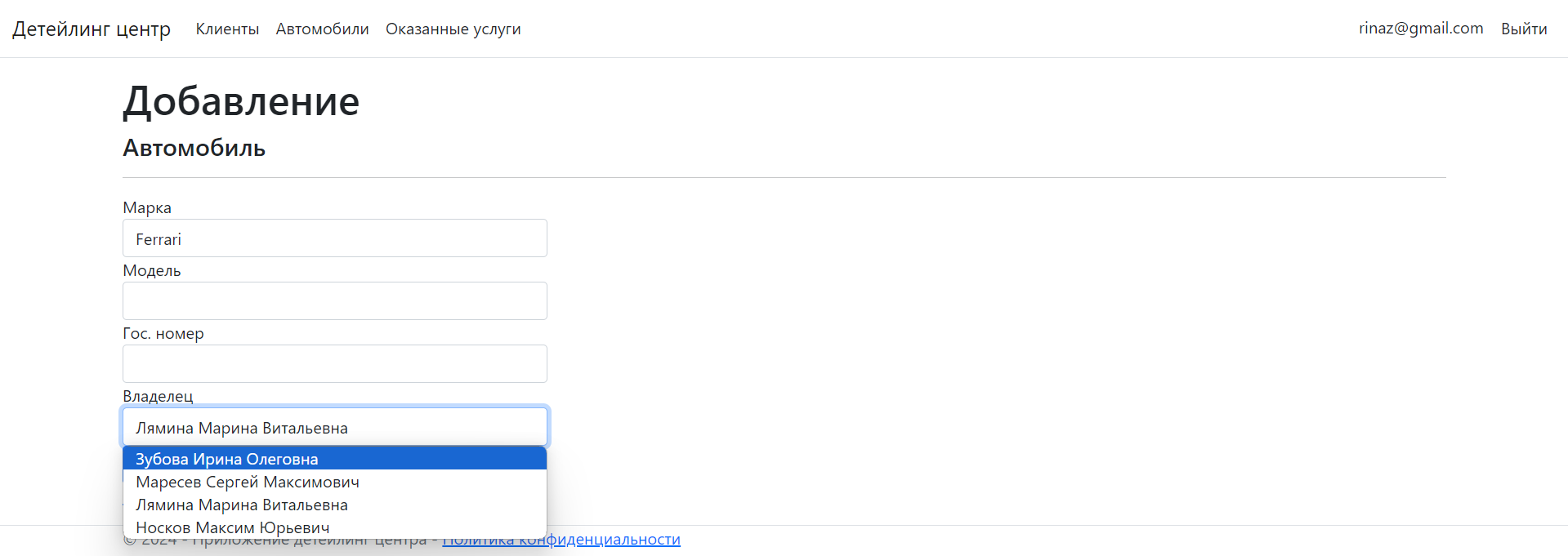


Рисунок 23 – Страница добавления автомобиля

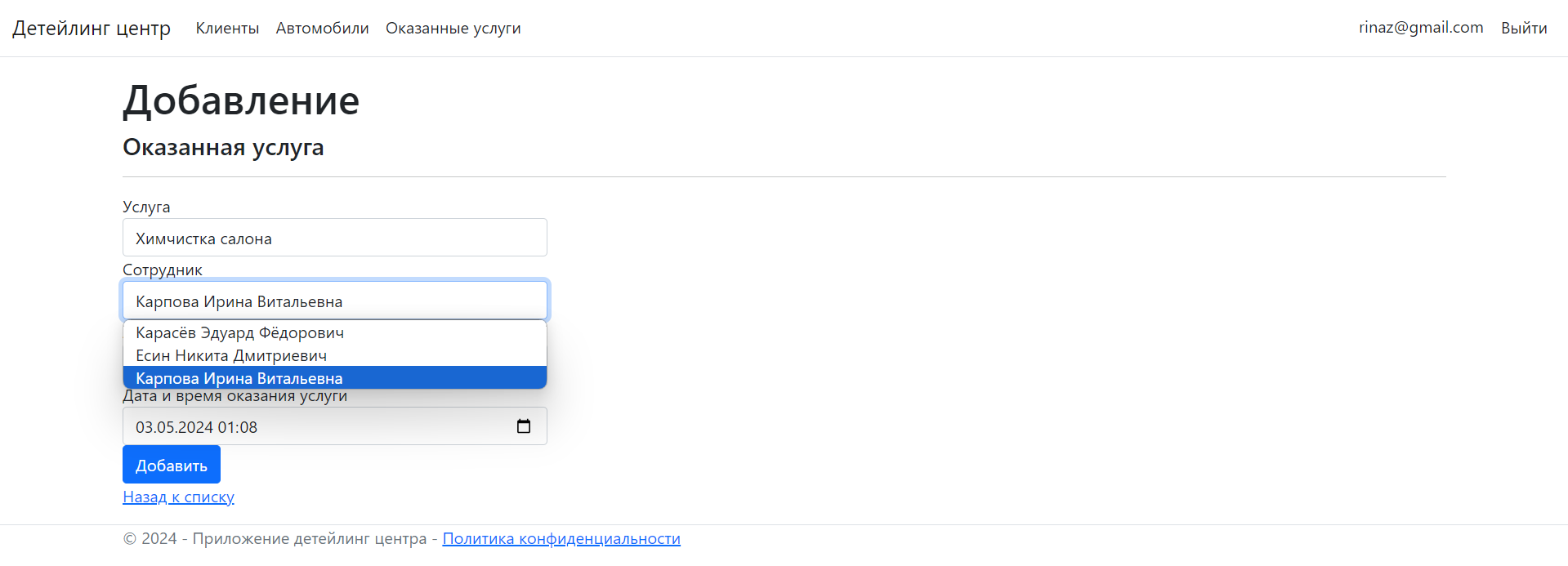


Рисунок 24 – Страница добавления оказанной услуги

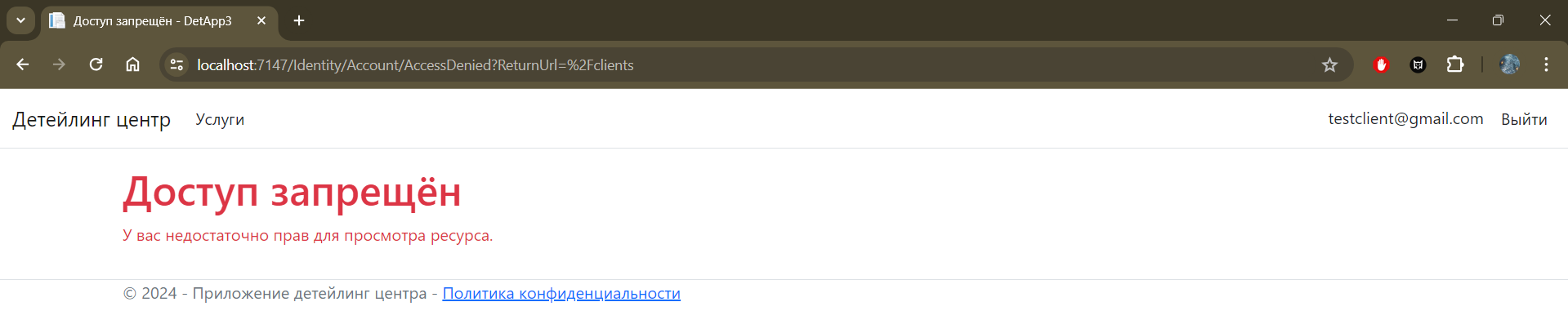


Рисунок 25 – Проверка настройки разграничения доступа

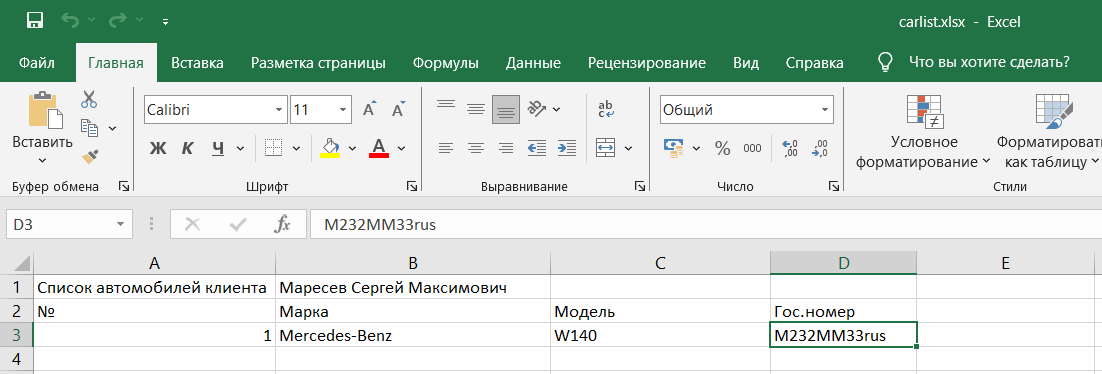


Рисунок 26 – Полученный отчёт «Список автомобилей клиента»

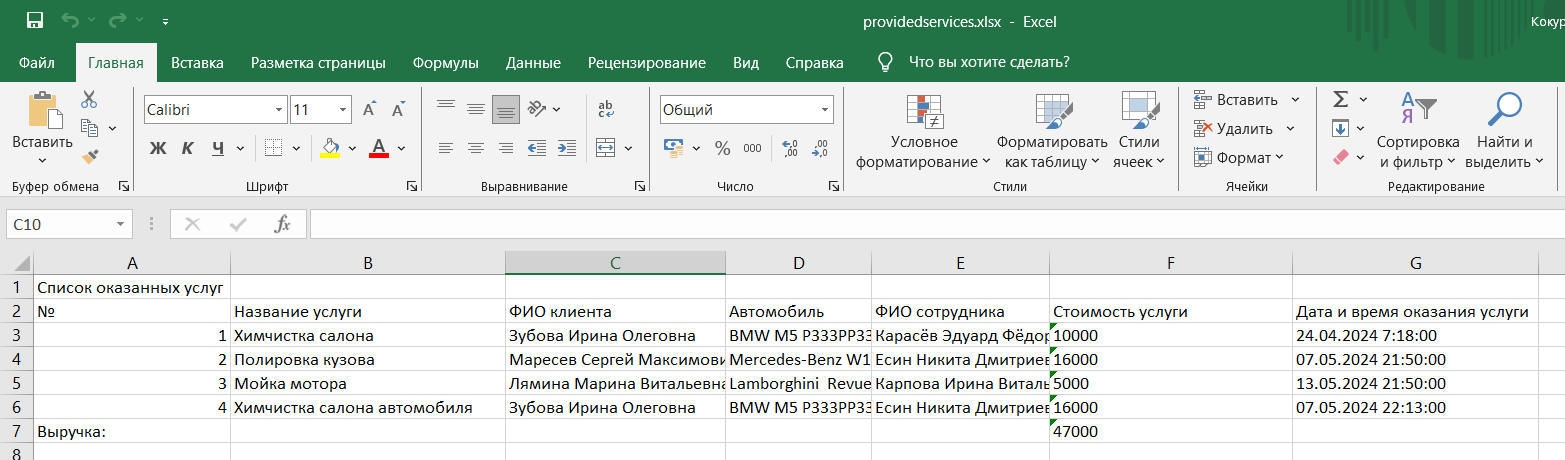


Рисунок 27 – Полученный отчёт «Список оказанных услуг и выручка»

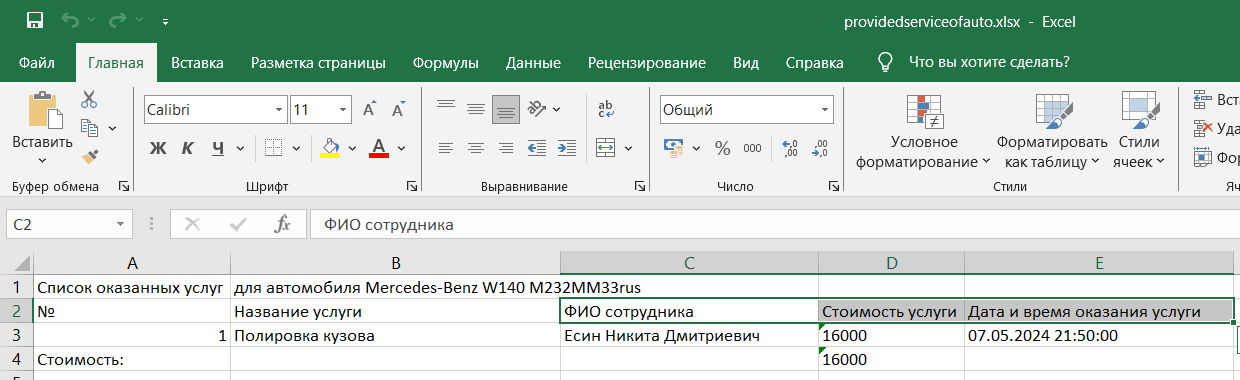


Рисунок 28 – Полученный отчёт «Список оказанных услуг и их стоимость» для конкретного автомобиля