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 Разработка приложений для мобильных операционных систем

Тема АИС «Таксопарка»

Руководитель

Белякова А.С

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Носков М.Ю

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

В данной курсовой работе необходимо было спроектировать приложение таксопарка. В качестве средств разработки базы данных была использована среда Android Studio. Язык разработки: Kotlin.

In this course work, it was necessary to design a taxi company application. The Android Studio environment was used as a database development tool. Development language: Kotlin.

Содержание

Введение………………………………………………………………...6

1. Анализ технического задания……………………………….………8

2. Разработка моделей данных ………………………………………...10 3. Разработка и реализация АИС............................……………….….15

4. Тестирование……………………...……………………………….....29

Заключение…………………………………….………………….........31

Список используемой литературы……………………….........………..32

Приложение 1………………………………………......………………...34

Приложение 2…………………………………......……………………...36

Приложение 3. Снимки окон программы................................................37

Введение

Основой жизнеобеспечения любого города является пассажирский транспорт. Если говорить о массовом пассажирском транспорте, таком как маршрутный транспорт (автобусы, троллейбусы), то основные критерии контроля и управления так или иначе связаны с конкретным маршрутом, «пиковыми» нагрузками на маршрут или его участки. Однако стоит уделить отдельное внимание такому виду транспорта как такси, где нет конкретного маршрута следования, остановочных пунктов, расписания и т.п.

Любой программный продукт, призванный автоматизировать систему управления службой такси, является узким, специализированным отраслевым программным продуктом. Это связано не только со спецификой такого вида транспорта как такси, но и с наличием «посредника» в оказании транспортной услуги - диспетчера.

Автоматизированная информационная система (АИС) таксопарка представляет собой комплексное решение, направленное на оптимизацию процессов управления такси. Система обеспечивает хранение и обработку данных о водителях, автомобилях и заказах, что позволяет диспетчерам эффективно управлять ресурсами и повышать качество обслуживания клиентов. В условиях растущей конкуренции на рынке такси внедрение АИС становится необходимостью для повышения эффективности работы.

Автоматизированные информационные системы становятся неотъемлемой частью современного бизнеса, и таксопарки не являются исключением. АИС таксопарка обеспечивает эффективное управление ресурсами, включая водителей, автомобили и заказы. В условиях растущей конкуренции и потребности в оптимизации процессов, создание такой системы на платформе Android с использованием языка Kotlin представляет собой актуальную задачу.

Цель данной курсовой работы — разработка АИС «Таксопарка» на платформе Android с использованием языка программирования Kotlin. Система будет включать в себя функционал для диспетчеров, позволяющий отслеживать состояние автомобилей, управлять заказами и анализировать деятельность водителей. Важным аспектом является также обеспечение надежности и удобства использования приложения.

Анализ технического задания.

Для успешной реализации проекта необходимо детально рассмотреть каждое требование технического задания.

1. Функционал:

- Сведения о водителях: регистрация, авторизация, изменение данных. Водители должны иметь возможность легко зарегистрироваться и войти в систему. Это обеспечит безопасность их данных.

- Сведения об автомобилях: учет автомобилей на линии и в ремонте, история выполнения заказов. Приложение будет отслеживать состояние автомобилей — на линии или в ремонте. Это поможет водителям и администраторам управлять автопарком.

- Сведения о заказах и бронированиях: регистрация новых заказов, подтверждение бронирований, мониторинг статуса заказов. Приложение будет показывать текущий статус заказов, что позволит водителям планировать свою работу.

- Возможность получения информации о количестве выполненных заказов каждым водителем.

2. Операционная система и языки программирования:

Android: Приложение будет разработано для платформы Android, что обеспечит доступ к широкой аудитории пользователей. Kotlin: Использование языка Kotlin позволит создать современное и безопасное приложение с лаконичным кодом.

3. Документация:

- Документирование кода с помощью инструмента Dokka. Данный инструмент предоставляет структурированный и удобный формат документации, который облегчит понимание и поддержку разрабатываемого приложения.

4. Система контроля версий:

- Размещение исходного кода в репозитории системы контроля версий GitHub. Это обеспечит централизованное управление версиями, возможность совместной работы над проектом и легкий доступ к истории изменений.

5. Тестирование:

- Модульное тестирование разработанного приложения. Этот этап гарантирует корректность отдельных компонентов системы и их взаимодействие между собой.

- Функциональное тестирование на не менее трех виртуальных устройствах. Проверка работоспособности приложения на различных конфигурациях устройств и операционных систем.

- Автоматизированное тестирование интерфейса. Это позволит быстро и эффективно проверять пользовательский интерфейс на наличие ошибок и соответствие требованиям дизайна.

Реализация каждого этапа должна быть тщательно спланирована и согласована с заказчиком для достижения максимальной эффективности и качества разрабатываемой системы.

2. Разработка моделей данных

2.1 Концептуальная модель данных

На первом этапе разработки автоматизированной информационной системы создается концептуальная модель, которая визуализирует предметную область системы. Хотя концептуальная модель не предназначена для прямой реализации, все таблицы должны быть нормализованы для использования в реляционной базе данных. Следующим шагом является создание логической модели, которая аналогична концептуальной, но более понятна для бизнес-пользователей и учитывает все аспекты предметной области.

Затем происходит разработка физической модели данных, определяющей структуру и доступ к данным на уровне хранения. В физической модели учитываются детали, такие как названия таблиц, типы данных и ключи. Для создания концептуальной модели выделяются сущности, после чего формируется модель. Этот процесс необходим для дальнейшего проектирования базы данных. Обладая всей необходимой информацией, можно составить окончательную концептуальную модель, которая покажет взаимосвязи между таблицами и данные, которые они содержат.

Имея всю необходимую информацию, можно легко составить окончательную концептуальную модель, которая покажет, как таблицы будут взаимодействовать (связи между объектами) и какие данные они будут хранить. Результат отражён в приложении 1 (рисунок 1).

2.2 Логическая модель данных

Логическая модель данных представляет собой описание структуры данных, их связей и основных характеристик. Она помогает определить, какие данные будут храниться в базе данных, как они будут взаимосвязаны и как будут использоваться в прикладных системах.

Ключевые аспекты логической модели данных включают:

1. Сущности и их атрибуты: описание основных объектов (например, сотрудники, заказы, продукты) и их характеристик.

2. Связи между сущностями: определение взаимосвязей между различными объектами, таких как один к одному, один ко многим, многие ко многим.

3. Ограничения целостности данных: установление правил для обеспечения целостности данных, например, уникальность значений, обязательность заполнения полей.

4. Нормализация данных: организация данных для избежания избыточности и неоднозначности информации.

5. Учет бизнес-логики: учитывание особенностей бизнес-процессов и правил, которые должны отражаться в структуре данных.

Логическая модель данных фокусируется на описании структуры данных и их взаимосвязей в предметной области, независимо от конкретной реализации базы данных и языка запросов.

На рисунке 2 (Приложение 1) представлена логическая модель для информационной системы охранного агентства. Логическая модель данных представляет собой описание структуры данных и их взаимосвязей в информационной системе охранного агентства. Она фокусируется на сущностях, их атрибутах, связях и бизнес-логике без учета технических деталей хранения.

Физическая модель данных, с другой стороны, определяет способ хранения данных в базе данных, учитывая типы данных, индексы, ограничения целостности и другие технические аспекты. Например, для таблицы "Сотрудники" в физической модели определяются типы данных для каждого атрибута, устанавливаются первичные ключи, создаются индексы для оптимизации запросов.

Таким образом, логическая модель описывает структуру данных на уровне предметной области, в то время как физическая модель определяет их хранение и доступ к данным в реальной базе данных.

Физическая модель данных, представленная на рисунке 3 в приложении 1, составлена на основе логической модели, где уже отражены и учтены все основные аспекты исследуемой предметной области. После проектирования этой модели будет осуществляться непосредственно создание базы данных и ее таблиц.

3. Проектирование работы системы

В данной главе рассматривается проектирование работы системы автоматизированной информационной системы охранного агентства. Для начала, необходимо определить основные компоненты системы и их взаимодействие.

При проектировании работы системы автоматизированной информационной системы охранного агентства необходимо 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 баз данных, таких как SQL Server или MySQL.

Первоначальным этапом проектирования является разработка моделей данных. Для учета охраняемых объектов, владельцев, договоров, услуг и сотрудников необходимо создать соответствующие сущности и связи между ними. К примеру, модель данных может включать таблицы для клиентов, объектов охраны, договоров, услуг и сотрудников, а также предусматривать связи между ними, такие как связь "один ко многим" между клиентами и договорами.

Далее следует создание CRUD операций для администратора системы. Это позволит ему управлять данными о клиентах, объектах охраны, договорах и услугах. В рамках создания CRUD функционала важно предусмотреть возможности добавления, просмотра, обновления и удаления записей.

Для пользователя системы необходимо предусмотреть функционал регистрации в системе, просмотра заключенных договоров и возможность оставлять сообщения. 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 за определенный период времени, количество заключенных договоров, статистику по сотрудникам и статистику вызовов охраны.

Интерфейс приложения должен быть интуитивно 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 охраняемых объектов, владельцев, договоров, услуг и сотрудников. Для взаимодействия с базой данных будет использоваться SQL Server или MySQL.

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.

В заключение, проектирование работы системы автоматизированной информационной системы охранного агентства включает в себя проектирование базы данных, разработку пользовательского интерфейса, создание CRUD операций для администратора, реализацию функционала для пользователей и создание механизма генерации отчетов. Важно уделить внимание архитектуре приложения, безопасности данных и удобству использования для конечных пользователей.

4. Разработка и реализация системы

Ниже будет описан процесс проектирования и создания таблиц базы данных для АИС охранного агентства. Каждая таблица была разработана с учетом специфики данных и их взаимосвязей.

Всего было разработано пять таблиц. Каждая таблица содержит внешние ключи связывающие текущую таблицу с другими.

Таблица сотрудников агенства:

public class Employees

{

public int Id { get; set; }

[Display (Name = "ФИО")]

[Required(ErrorMessage = "Поле 'ФИО' обязательно для заполнения.")]

public string full\_name { get; set; }

[Display(Name = "Должность")]

[Required(ErrorMessage = "Поле 'Должность' обязательно для заполнения.")]

public string position { get; set; }

[Display(Name = "Дата поступления на работу")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime hire\_date { get; set; }

[Display(Name = "Зарплата")]

[Range(0, int.MaxValue, ErrorMessage = "Зарплата должна быть неотрицательным числом.")]

public int salary { get; set; }

[Display(Name = "Образование")]

public string education { get; set; }

[Display(Name = "Фото")]

Public string? Photo{get;set}

[ForeignKey("Secured\_ObjectsId")]

public int? Secured\_ObjectsId { get; set; }

[Display(Name = "Адрес охраняемого объекта")]

public Secured\_Objects? Secured\_Objects { get; set; }

}

Таблица "Сотрудники" представляет собой хранилище информации о персонале охранного агентства. Каждая запись содержит данные о конкретном сотруднике, включая его ФИО, должность, дату поступления на работу, зарплату, образование, а также ссылку на фотографию сотрудника. Кроме того, таблица содержит внешний ключ, связывающий сотрудника с объектом, на котором он работает.

Структура:

- Id: Уникальный идентификатор сотрудника.

- full\_name: ФИО сотрудника.

- position: Должность, занимаемая сотрудником.

- hire\_date: Дата поступления сотрудника на работу.

- salary: Зарплата сотрудника.

- education: Образование сотрудника.

- Photo: Ссылка на фотографию сотрудника.

- Secured\_ObjectsId: Внешний ключ, связывающий сотрудника с объектом, на котором он работает.

- Secured\_Objects: Ссылка на объект безопасности, на котором работает сотрудник.

Реализация:

Класс Employees представляет модель данных для таблицы "Сотрудники". В нем определены свойства, соответствующие каждому столбцу таблицы, а также атрибуты для валидации данных. Поле Secured\_ObjectsId используется для хранения внешнего ключа, связывающего сотрудника с объектом безопасности. Поле Secured\_Objects представляет собой навигационное свойство, позволяющее получить доступ к объекту безопасности, на котором работает сотрудник.

Данная таблица позволяет хранить и управлять информацией о персонале охранного агентства. С помощью интерфейса пользователя можно добавлять новых сотрудников, редактировать их данные, а также просматривать информацию о каждом сотруднике, включая его фотографию и привязку к объекту безопасности.

Таблица "Договоры"

public class Contracts

{

public int Id { get; set; }

[Display(Name = "Сумма")]

public int summ\_contracts { get; set; }

[Display(Name = "Дата завершения договора")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime date\_completion { get; set; }

[Display(Name = "Дата заключения договора")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime date\_conclution { get; set; }

[Display(Name = "Серийный номер контракта")]

public int contract\_number { get; set; }

[Display(Name = "Вид услуги")]

public string Service\_Type { get; set;}

public virtual ICollection<Services> Services { get; set; }

public virtual ICollection<Customer> Customer { get; set; }

public virtual ICollection<Secured\_Objects> Secured\_Objects { get; set; }

public Contracts()

{

Services = new List<Services>();

Customer = new HashSet<Customer>();

Secured\_Objects = new List<Secured\_Objects>();

}

}

Описание:

Таблица "Договоры" представляет собой хранилище информации о заключенных договорах между охранным агентством и его клиентами. В каждой записи содержатся данные о сумме договора, датах его заключения и завершения, серийном номере контракта, а также виде услуги, предоставляемых клиенту в рамках данного договора. Таблица также содержит связи с другими таблицами для учета информации о предоставленных услугах, клиентах и охраняемых объектах.

Структура:

- Id: Уникальный идентификатор договора.

- summ\_contracts: Сумма договора.

- date\_completion: Дата завершения действия договора.

- date\_conclution: Дата заключения договора.

- contract\_number: Серийный номер контракта.

- Service\_Type: Вид услуги, предоставляемых в рамках договора.

- Services: Коллекция предоставленных услуг.

- Customer: Коллекция клиентов, связанных с данным договором.

- Secured\_Objects: Коллекция охраняемых объектов, связанных с данным договором.

Реализация:

Класс Contracts представляет модель данных для таблицы "Договоры". В нем определены свойства, соответствующие каждому столбцу таблицы, а также коллекции для хранения связанных сущностей, таких как предоставленные услуги, клиенты и охраняемые объекты. В конструкторе класса инициализируются эти коллекции.

Данная таблица позволяет хранить информацию о заключенных договорах между охранным агентством и его клиентами. Через интерфейс системы можно добавлять новые договоры, редактировать их данные, а также просматривать информацию о каждом договоре, включая его сумму, даты, серийный номер и связанные услуги, клиенты и охраняемые объекты.

Таблица "Клиенты"

public class Customer

{

public int Id { get; set; }

[Display(Name = "Имя")]

[Required(ErrorMessage = "Поле 'Имя' обязательно для заполнения.")]

public string Name { get; set; }

[Display(Name = "Адрес")]

[Required(ErrorMessage = "Поле 'Адрес' обязательно для заполнения.")]

public string Address { get; set; }

[Display(Name = "ID договора")]

[ForeignKey("ContractsId")]

public int? ContractsId { get; set; }

public Contracts? Contracts { get; set; }

[Display(Name = "Серийный номер клиента")]

public int account\_number { get; set; }

[RegularExpression(@"^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$", ErrorMessage = "Номер счета должен содержать 10 цифр.")]

[Display(Name = "Телефон")]

public string phone { get; set; }

}

Таблица "Клиенты" представляет собой хранилище информации о клиентах охранных услуг. В каждой записи содержатся данные о имени клиента, его адресе, контактном телефоне, а также ссылка на соответствующий договор, заключенный с клиентом. Также в таблице присутствует поле с серийным номером клиента, которое может использоваться для уникальной идентификации клиента.

Структура:

- Id: Уникальный идентификатор клиента.

- Name: Имя клиента.

- Address: Адрес клиента.

- ContractsId: Внешний ключ, связывающий клиента с соответствующим договором.

- Contracts: Ссылка на договор, заключенный с клиентом.

- account\_number: Серийный номер клиента.

- phone: Контактный телефон клиента.

Реализация:

Класс Customer представляет модель данных для таблицы "Клиенты". В нем определены свойства, соответствующие каждому столбцу таблицы, а также внешний ключ и навигационное свойство для связи с договором. Для уникальной идентификации клиента предусмотрено поле account\_number. Также присутствует валидация данных, гарантирующая заполнение обязательных полей и соответствие формату телефонного номера.

Данная таблица позволяет хранить информацию о клиентах охранных услуг, их контактных данных и связанных с ними договорах. Через интерфейс системы можно добавлять новых клиентов, редактировать их данные, просматривать информацию о каждом клиенте и его договорах.

Таблица "Охраняемые объекты"

public class Secured\_Objects

{

public int Id { get; set; }

[Display(Name = "Адрес")]

[Required(ErrorMessage = "Поле 'Адрес' обязательно для заполнения.")]

public string adress { get; set; }

[Display(Name = "Серийный номер объекта")]

[Required(ErrorMessage = "Поле 'Серийный номер объекта' обязательно для заполнения.")]

public int account\_number { get; set; }

[Display(Name = "Название объекта")]

[Required(ErrorMessage = "Поле 'Название объекта' обязательно для заполнения.")]

public string full\_name { get; set; }

[Display(Name = "Телефон")]

[Required(ErrorMessage = "Поле 'Телефон' обязательно для заполнения.")]

[RegularExpression(@"^\d{10}$", ErrorMessage = "Телефон должен состоять из 10 цифр.")]

public int phone { get; set; }

[Display(Name = "Вид услуги")]

[Required(ErrorMessage = "Поле 'Вид услуги' обязательно для заполнения.")]

public string service\_type { get; set; }

public int? ContractsId { get; set; }

[ForeignKey("ContractsId")]

[Display(Name = "ID контракта")]

public Contracts? contracts { get; set; }

public virtual ICollection<Employees> Employeess { get; set; }

public Secured\_Objects()

{

Employeess = new List<Employees>();

}

}

Таблица "Охраняемые объекты" содержит информацию об объектах, которые охраняет охранное агентство. Каждая запись в таблице содержит данные об адресе объекта, его названии, контактном телефоне, серийном номере, виде предоставляемых услуг, а также ссылке на соответствующий договор об оказании услуг.

Структура:

- Id: Уникальный идентификатор объекта безопасности.

- adress: Адрес объекта.

- account\_number: Серийный номер объекта.

- full\_name: Название объекта.

- phone: Контактный телефон объекта.

- service\_type: Вид оказываемых услуг.

- ContractsId: Внешний ключ, связывающий объект безопасности с соответствующим договором.

- contracts: Ссылка на договор об оказании услуг.

- Employeess: Коллекция сотрудников, назначенных на охраняемый объект.

Реализация:

Класс Secured\_Objects представляет модель данных для таблицы "Охраняемые объекты". В нем определены свойства, соответствующие каждому столбцу таблицы, а также внешний ключ и навигационное свойство для связи с договором об оказании услуг. Для каждого объекта безопасности в таблице хранится коллекция сотрудников, которые назначены на его охрану.

Данная таблица позволяет хранить информацию об охраняемых объектах и связанных с ними договорах. Через интерфейс системы можно добавлять новые объекты безопасности, редактировать их данные, просматривать информацию о каждом объекте и связанных с ним договорах, а также назначать сотрудников на охрану объектов.

Таблица "Услуги"

public class Services

{

public int Id { get; set; }

[Display(Name = "Вид")]

[Required(ErrorMessage = "Поле 'Вид' обязательно для заполнения.")]

public string Name { get; set; }

[Display(Name = "Цена")]

[Required(ErrorMessage = "Поле 'Цена' обязательно для заполнения.")]

[RegularExpression(@"^\d+(\.\d{1,2})?$", ErrorMessage = "Цена должна быть числом с двумя цифрами после точки (если есть).")]

public string Cost { get; set; }

public int? ContractId { get; set; }

[ForeignKey("ContractsId")]

[Display(Name = " ID контракта")]

public Contracts? Contracts { get; set; }

}

Таблица "Услуги" содержит информацию о предоставляемых охранной агентством услугах. Каждая запись в таблице содержит данные о виде услуги и их стоимости, а также ссылке на соответствующий договор, в рамках которого оказываются эти услуги.

Структура:

- Id: Уникальный идентификатор услуги.

- Name: Название услуги.

- Cost: Цена услуги.

- ContractId: Внешний ключ, связывающий услугу с соответствующим договором.

- Contracts: Ссылка на договор, в рамках которого предоставляются услуги.

Реализация:

Класс Services представляет модель данных для таблицы "Услуги". В нем определены свойства, соответствующие каждому столбцу таблицы, а также внешний ключ и навигационное свойство для связи с договором об оказании услуг. Для валидации данных предусмотрено обязательное заполнение названия и цены услуги, а также проверка формата цены.

Данная таблица позволяет хранить информацию о предоставляемых охранной агентством услугах и связанных с ними договорах. Через интерфейс системы можно добавлять новые услуги, редактировать их данные, просматривать информацию о каждой услуге и связанных с ней договорах.

Одной из целей проекта было создание функционала регистрации и авторизации пользователей, что 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. Для расширения модели пользователя мы добавили дополнительные свойства, такие как имя и фамилия:

public class ApplicationUser : IdentityUser

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

Далее была создана миграция и обновление базы данных.

Настройка регистрации пользователей

Форма регистрации находится в файле Register.cshtml. Она была расширена, чтобы включить поля для ввода имени и фамилии:

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

<input asp-for="Input.FirstName" class="form-control" placeholder="First name" />

<label asp-for="Input.FirstName">First name</label>

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

</div>

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

<input asp-for="Input.LastName" class="form-control" placeholder="Last name" />

<label asp-for="Input.LastName">Last name</label>

<span asp-validation-for="Input.LastName" 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 охранного агентства. Настройка и интеграция Identity позволили создать гибкий и расширяемый механизм аутентификации и авторизации, обеспечивающий удобство и безопасность для пользователей.

Данный функционал представлен в приложении 3 (рис.7, рис.8).

Одна из задач курсового проекта было внедрение механизма создания отчетов с использованием сторонней библиотеки EPPlus для работы с файлами Excel. Для обеспечения безопасности и управления доступом к системе была реализована регистрация и авторизация пользователей с использованием Microsoft Identity. В проекте использовалась ASP.NET Core Identity для создания и управления учетными записями.

Создание отчетов в формате Excel было реализовано с использованием библиотеки EPPlus. Эта библиотека позволяет легко создавать и модифицировать файлы Excel, что значительно упрощает процесс генерации отчетов.

В рамках курсового проекта была разработана и внедрена система для охранного агентства, включающая регистрацию и авторизацию пользователей, а также генерацию 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. Этот проект демонстрирует интеграцию различных технологий для создания функциональной и удобной информационной системы.

5. Тестирование функционала в АИС охранного агентства

В этой главе представлены результаты тестирования функционала базы данных АИС охранного агентства. Для тестирования использовались следующие таблицы с внешними ключами: Employees,Contracts, Customer, Secured\_Objects,

Services.

Таблица 1 – Тестирование разработанного программного продукта

|  |  |  |
| --- | --- | --- |
| Выполняемое действие | Ожидаемый результат | Полученный результат |
| Запуск приложения | Появление основной формы приложения. | Появление основной формы приложения. |
| Добавление записей в таблицы | Добавление записи в таблицу и отображение ее на форме | Записи были успешно добавлены в базу данных без ошибок. |
| Переходы между таблицами с помощью интерфейса | Корректные переходы между таблицами | Успешные переходы между моделями. |
| Нажатие кнопки  «Создать отчет» | Создание отчета и вывод диалогового окна для выбора сохранения файла | Успешное создание отчета и сохранение его на устройстве пользователя |
| Нажатие кнопки «Регистрация» | Открытие формы для регистрации с полями для заполнения | Открытие формы для регистрации с заполняемыми полями |
| Регистрация пользователя | Регистрация пользователя и добавление его в базу данных | Успешная регистрация пользователя |
| Нажатие кнопки  «Авторизация» | Открытие формы для аваторизации с полями для заполнения | Открытие формы для авторизации с заполняемыми полями |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Нажатие кнопки  «Обновить данные» | Открытие формы для обновления записи | Открытие формы для обновления записи |
| Авторизация пользователя | Авторизация пользователя и вход в профиль | Успешная авторизация пользователя |
| Нажатие кнопки  «Удалить» на форме таблицы | Удаление записи из базы данных а также удаление с веб страницы | Успешное записи из базы данных а также удаление с веб страницы |
| Нажатие кнопки  «Детали» на форме таблицы | Переход на новую страницу с данными о конкретной записи | Успешный переход на новую страницу с определенными данными о конкретной записи |
| Нажатие кнопки  «Редактировать» на форме таблицы | Переход на новую страницу с заполняемыми полями и отображением фотографии | Успешный переход на новую страницу с заполняемыми полями и отображением фотографии |
| 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 приложения, позволяют сделать заключение о том, что созданное приложение соответствует заявленным требованиям.

Заключение

В ходе выполнения курсового проекта была разработана распределенная информационная система (ИС) для автоматизации деятельности охранного агентства с использованием технологии ASP.NET Core MVC. Данный программный продукт предназначен для ведения учета охраняемых объектов, их владельцев, договоров на охрану объектов и сопутствующих услуг, а также выручки за различные временные промежутки. Система также позволяет управлять сотрудниками фирмы, реализующими охрану объектов.

Основные функциональные возможности системы включают:

1. Управление данными клиентов, объектов охраны, договоров и услуг:

Администраторы имеют возможность выполнять CRUD (создание, чтение, обновление, удаление) операции для обработки данных клиентов, объектов охраны, договоров и охранных услуг.

2. Регистрация и авторизация пользователей:

Система реализует авторизацию пользователей с использованием Microsoft Identity. Доступный функционал и интерфейс зависят от роли пользователя (реализованы две роли: администратор и пользователь).

3. Адаптивный интерфейс:

Пользовательский интерфейс адаптирован для корректного отображения на различных устройствах и экранах.

4. Валидация данных:

Валидация данных осуществляется как на стороне клиента, обеспечивая корректность формата вводимых данных, так и на стороне сервера, проверяя соответствие данных логике работы системы.

5. Отчеты и статистика:

Система предоставляет возможность формирования сводных отчетов на основе информации из базы данных, включая объем выручки, количество договоров, статистику по сотрудникам и вызовам охраны. Отчеты могут быть сформированы в виде файлов Excel, что обеспечивает удобство их использования и анализа.

Таким образом, разработанная ИС охранного агентства является мощным инструментом для автоматизации деятельности агентства, улучшения качества обслуживания клиентов и оптимизации внутренних бизнес-процессов. Внедрение данной системы позволит охранному агентству повысить свою конкурентоспособность, предоставляя более качественные и оперативные услуги.

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

Приложение 1.

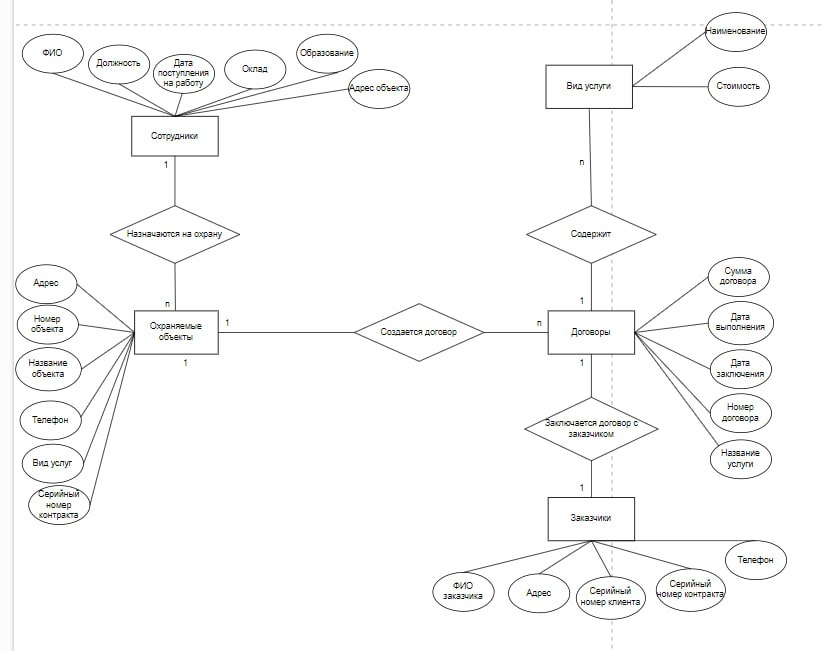


Рисунок 1 - Концептуальная модель данных

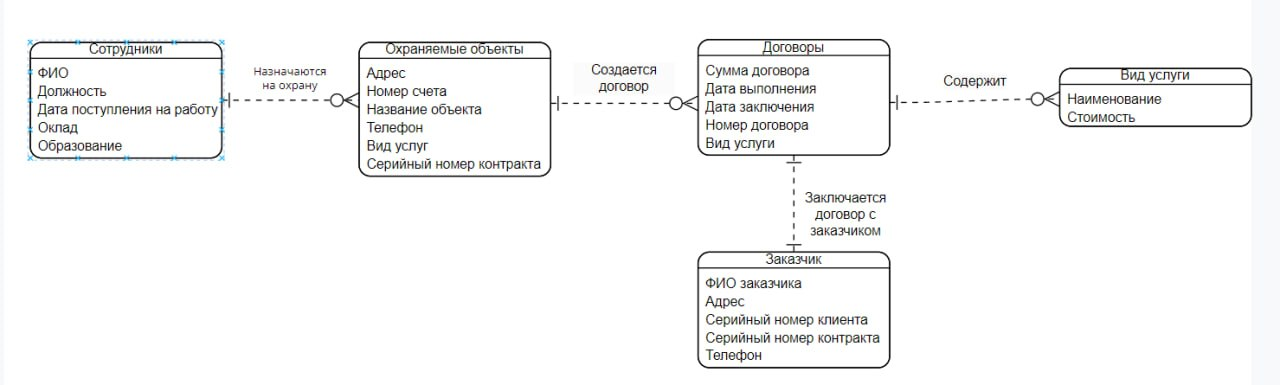


Рисунок 2 – Логическая модель данных

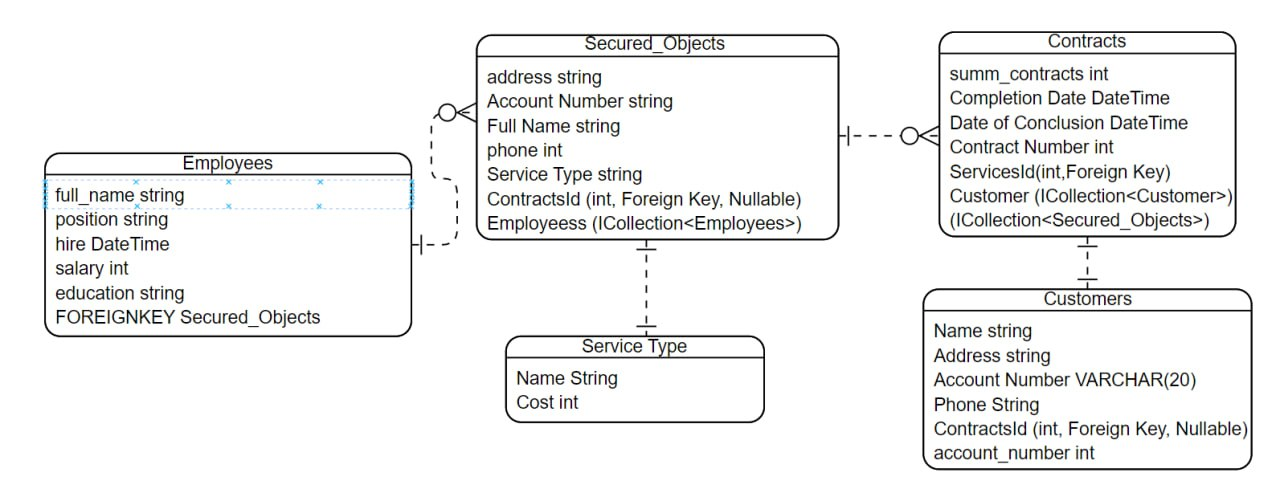


Рисунок 3 – Физическая модель данных

Приложение 2 . Программный код продукта

В процессе разработки программного продукта для автоматизации работы охранного агентства были проведены тесты, которые подтвердили соответствие всех функциональных требований. Выявленные ошибки были устранены, что повысило качество программы. В целом, программа успешно прошла тестирование и готова к использованию в работе охранного агентства. Разработанный программный код учитывает возможные проблемы пользователей и содержит меры по предотвращению ошибок. Также использованы унификации и внешние классы для уменьшения масштабности кода и сокращения времени выполнения программы.

Ссылка на репозиторий GitHub <https://github.com/MaxaDromka/SecuredBase.git>

Приложение 3. Снимки окон программы

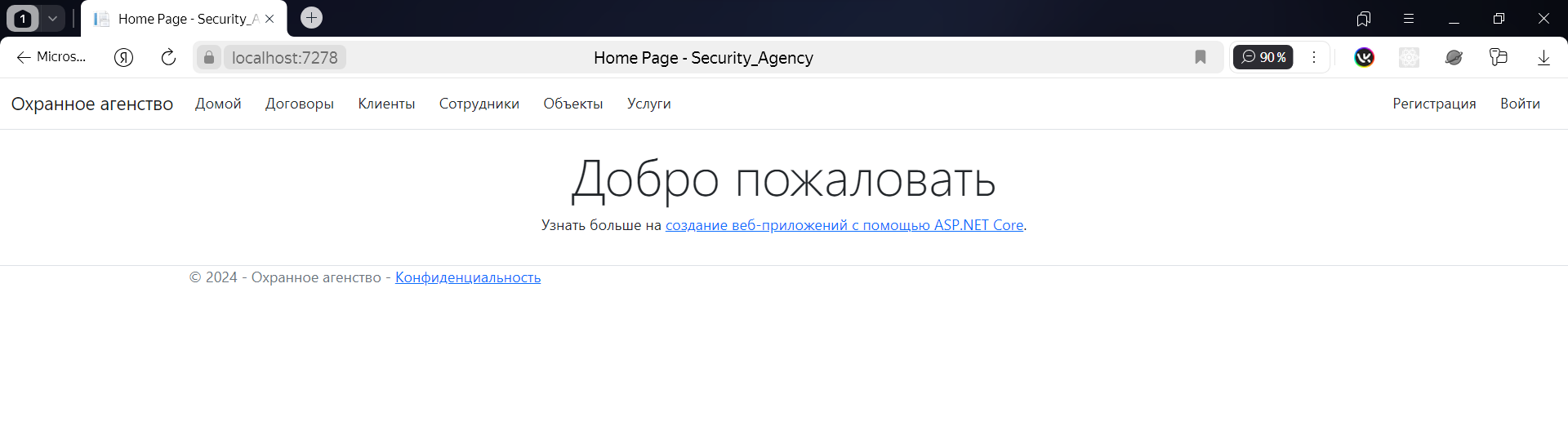


Рисунок 1 – Основное окно приложения

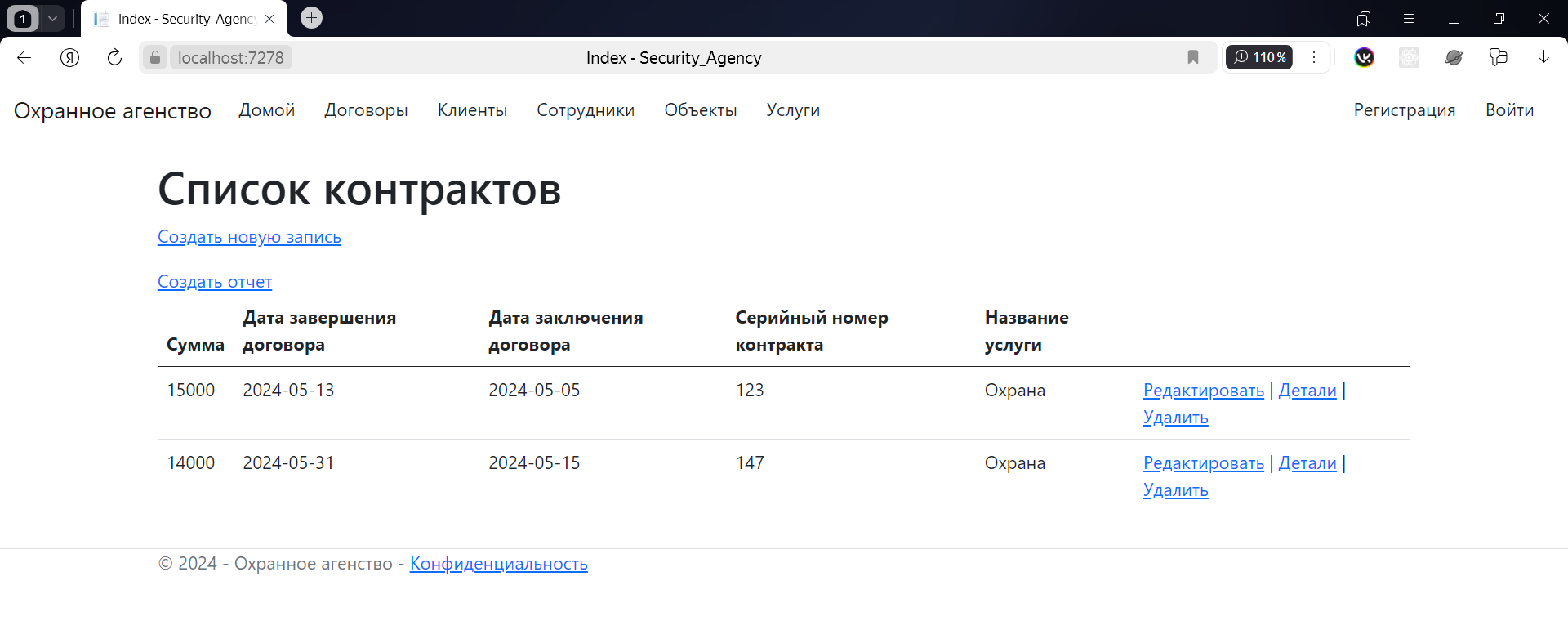


Рисунок 2 – Основное окно таблицы контрактов

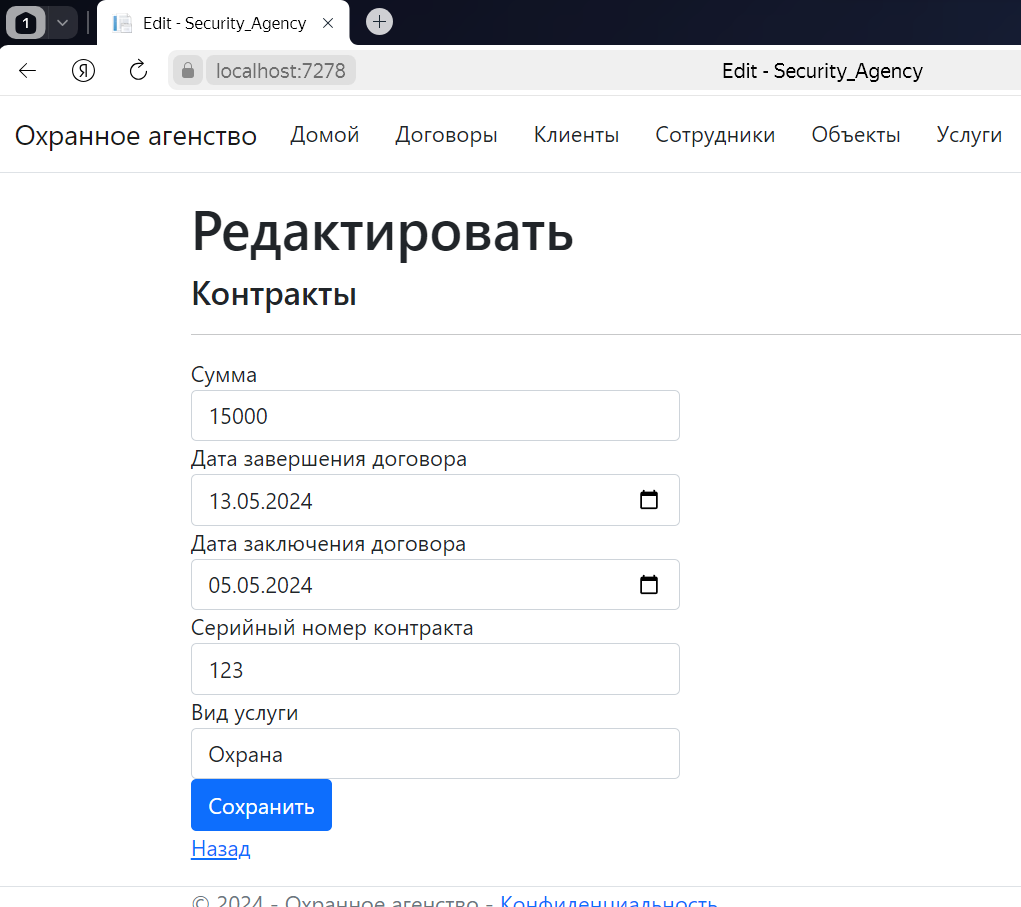


Рисунок 3 – Окно редактирования записи таблицы контрактов

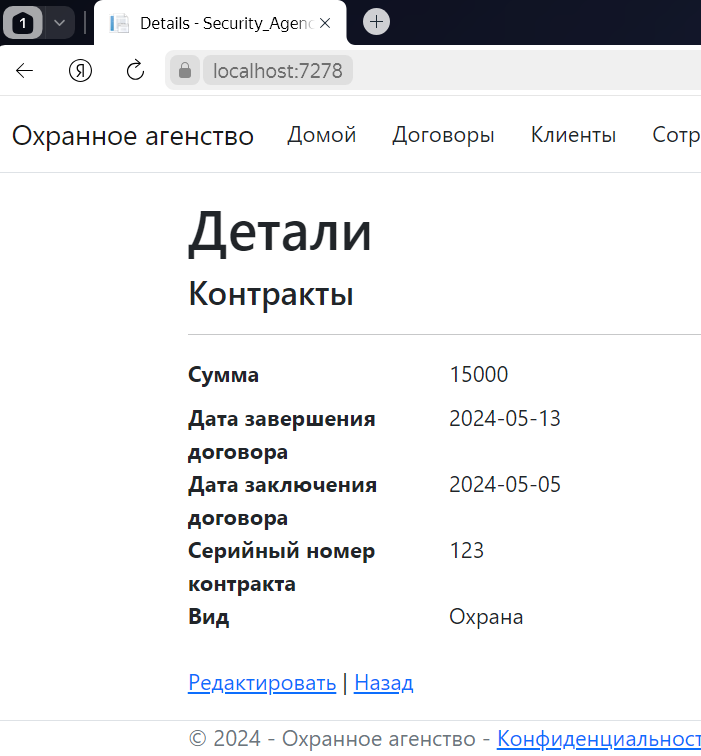


Рисунок 4 – страница для просмотра содержимого записи таблицы контрактов

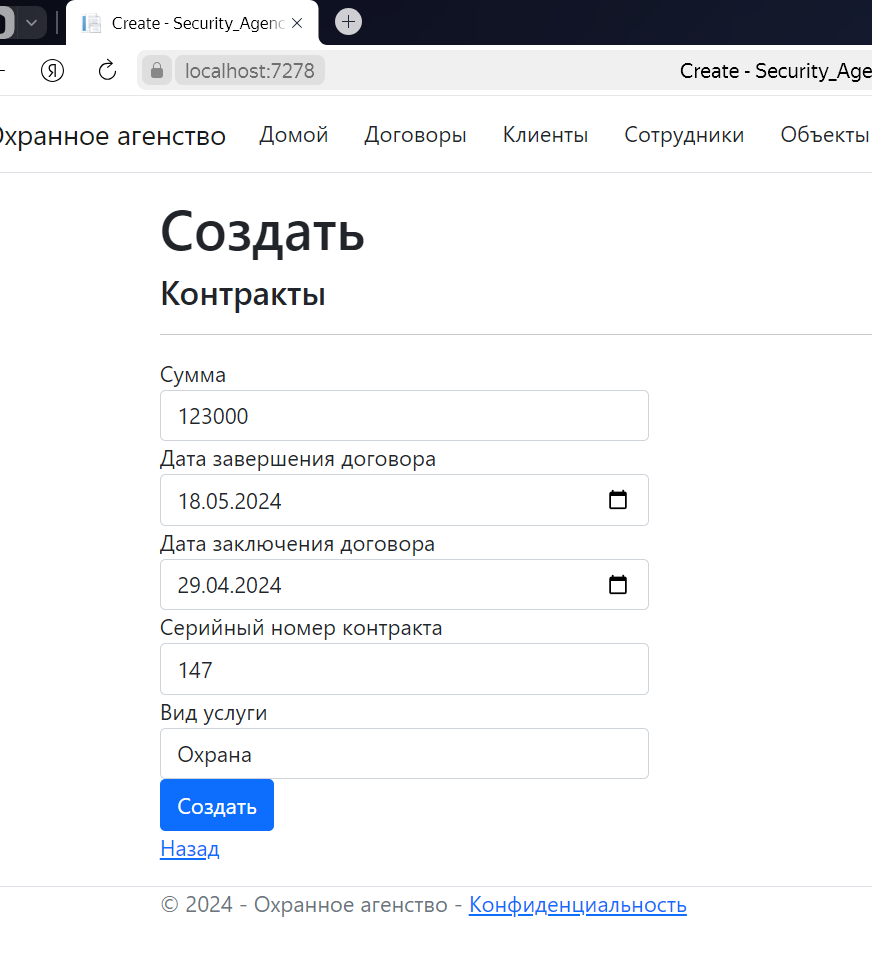


Рисунок 5 – Добавление записи в таблицу контрактов

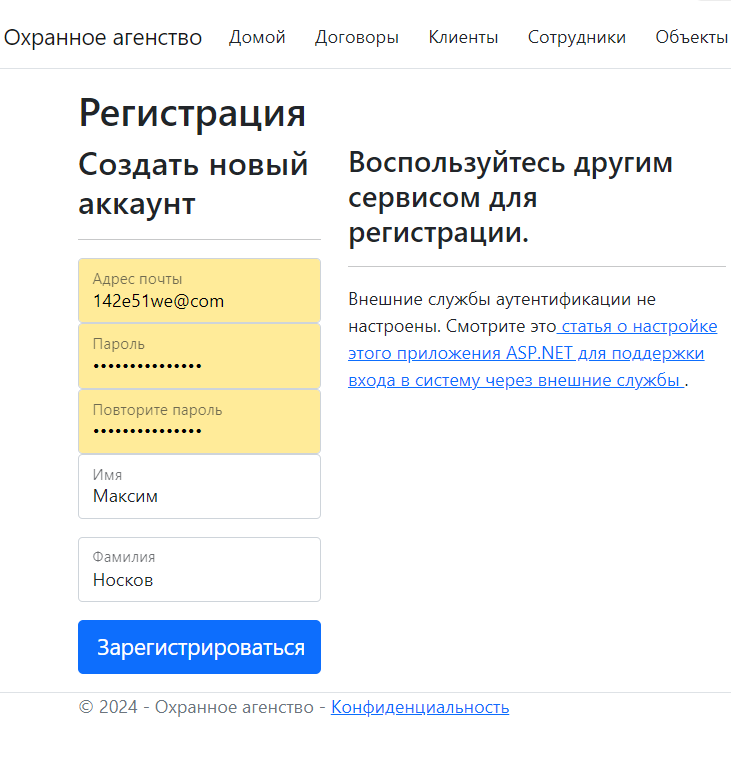


Рисунок 7 – регистрация пользователя в программе

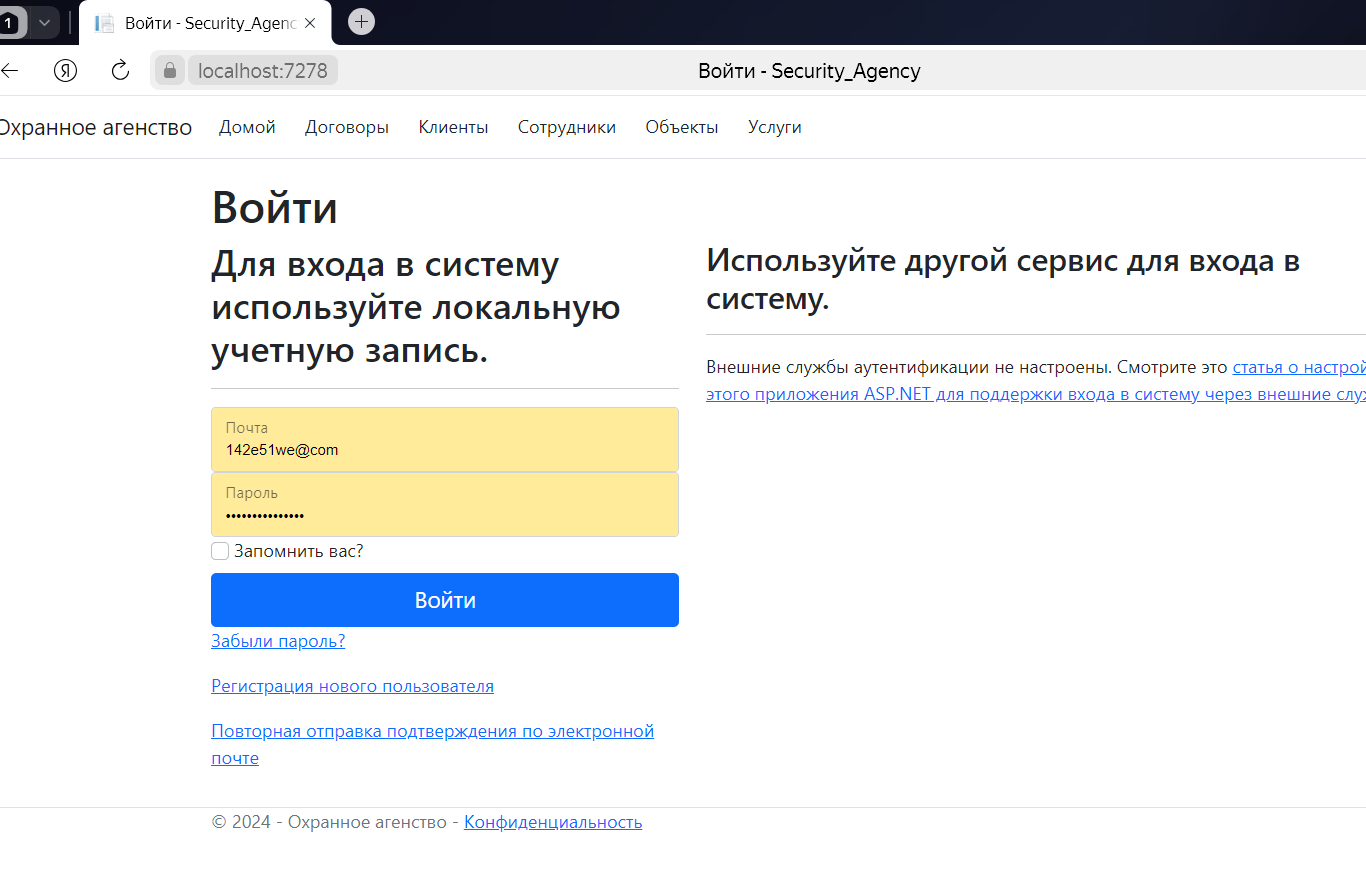


Рисунок 8 – авторизация пользователя

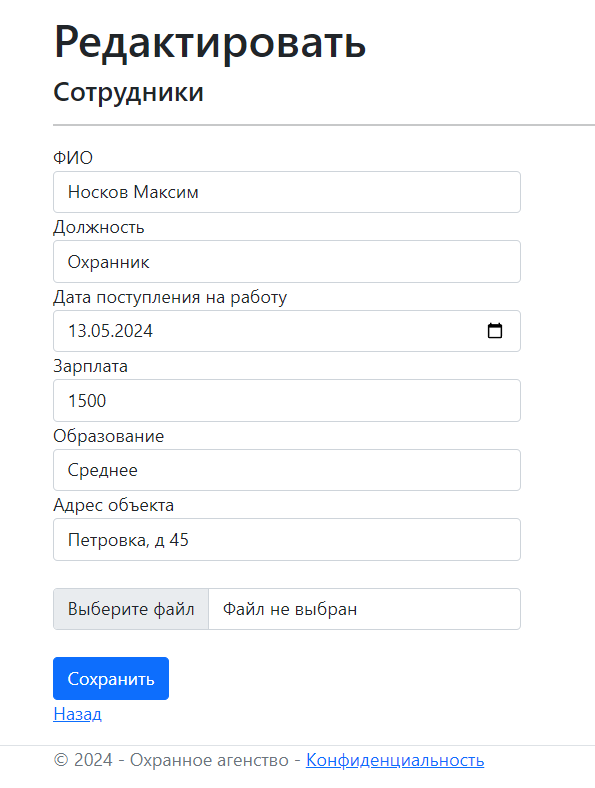


Рисунок 9 – загрузка фотографии на сервер

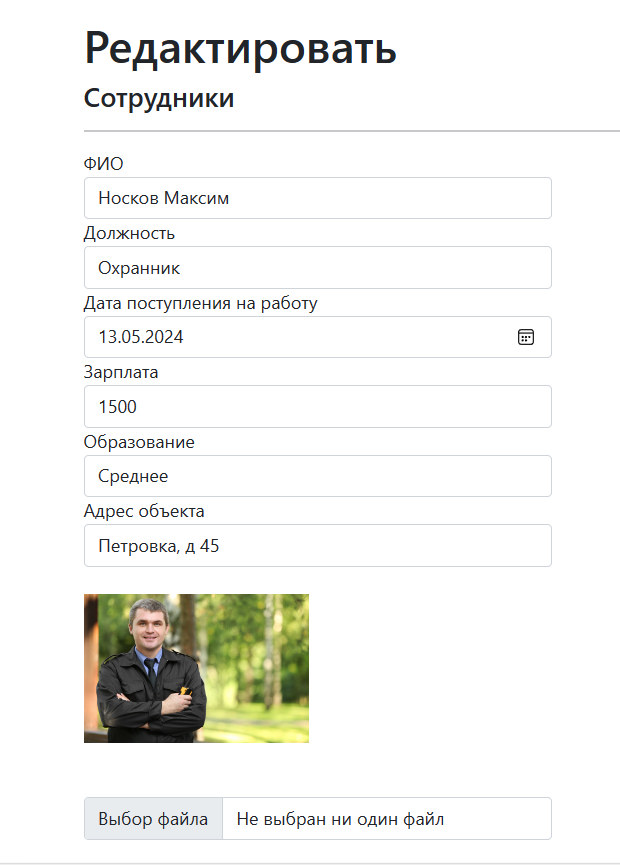


Рисунок 10 –полученная запись с файлом