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 «Строительная фирма»

Руководитель

Белякова А.С.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Линючев Н.А.

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

В данной курсовой работе необходимо было спроектировать информационную систему строительной фирмы. 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 construction company. 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 строительной фирмы на основе технологии ASP.NET Core MVC.

Строительная фирма играет ключевую роль в процессе создания различных строительных объектов - от жилых домов и коммерческих зданий до инфраструктурных сооружений. Она обладает необходимыми ресурсами, опытом и экспертизой для эффективной реализации проектов любой сложности. Кроме того, строительная фирма ответственна за соблюдение строительных норм и правил, безопасность на стройплощадке и контроль качества выполняемых работ.

Целями данной работы являются разработка моделей информационной системы строительной фирмы 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 строительной фирмы.

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 строительной фирме, занимающемся оказанием строительством объектов для заказчиков.

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.

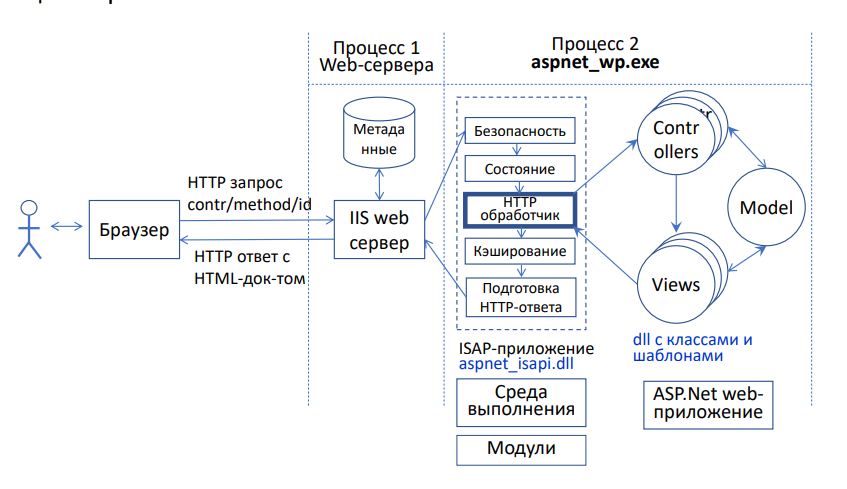


Рисунок 1 – Процесс работы APS.Net MVC

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.

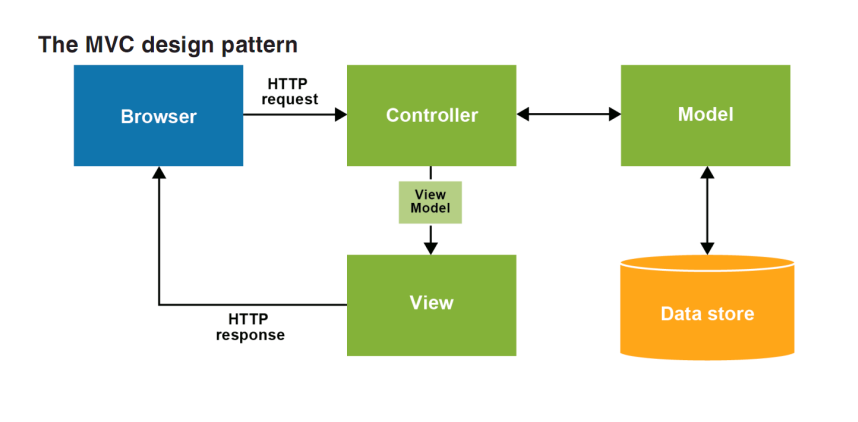


Рисунок 2 – Схема работы MVC-фреймворка

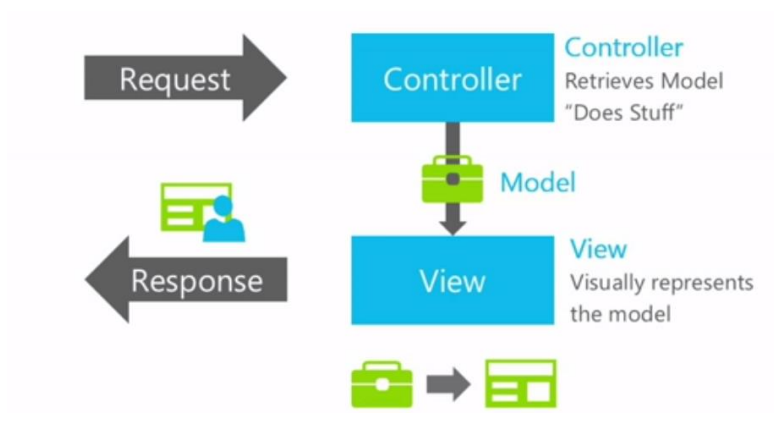


Рисунок 3 – Обработка запросов к MVC приложению

Программный продукт предназначен для автоматизации работы строительной фирмы. Администратор реализует CRUD методы обработки данных объектов строительства, исполнителей (строительных бригад), клиентов, управляет выполнением заказов строительными бригадами.

Клиент регистрируется в системе, формирует заявку на строительство объекта с указанием его характеристик, отслеживает статус выполнения строительства.

Должна быть возможность получения сводных отчетов на основе информации из БД: перечень заказов на строительство, перечень строительных объектов у выбранной бригада находящихся в стадии выполнения, перечень завершенных объектов, выручка от выполненных работ в целом, выбранной бригадой за любой период времени.

Дополнительные требования к разрабатываемой системе:

- Авторизация пользователей (использование Microsoft Identity);

- Интерфейс и доступный функционал зависит от роли пользователя (2 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 строительной фирмы 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 четырёх таблиц:

Первая таблица называется «Customers», в ней будет находиться информация о заказчиков строительной фирмы.

Таблица 1 – таблица «Customers».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| CustomerID | PRIMARY KEY, INTEGER, NOT NULL |
| Name | NVARCHAR(50), NOT NULL |
| LastName | NVARCHAR(100), NOT NULL |
| PhoneNamber | NVARCHAR(20), NOT NULL |

Вторая таблица называется «Foremens», в ней будет находиться информация о прорабах строительной фирмы.

Таблица 2 – таблица «Foremens».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ForemenID | PRIMARY KEY, INTEGER, NOT NULL |
| Name | NVARCHAR(50), NOT NULL |
| LastName | NVARCHAR(100), NOT NULL |
| PhoneNamber | NVARCHAR(100), NOT NULL |
| Qualification | NVARCHAR(100), NOT NULL |
| Specialization | NVARCHAR(100), NOT NULL |
| Skills | NVARCHAR(20), NOT NULL |

Третья таблица называется «Oobjects», в ней будет находиться информация об объектах строительной фирмы.

Таблица 3 – таблица «Oobjects».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| OobjectID | PRIMARY KEY, INTEGER, NOT NULL |
| Title | NVARCHAR(50), NOT NULL |
| Adress | NVARCHAR(130), NOT NULL |
| Type | NVARCHAR(50), NOT NULL |
| Status | NVARCHAR(50), NOT NULL |
| Photo | NVARCHAR(max), NULL |
| ForemenId | FOREIGN KEY, INTEGER, NULL |
| CustomerId | FOREIGN KEY, INTEGER, NULL |

Четвёртая таблица называется «Workers», в ней будет находиться информация о рабочих строительной фирмы.

Таблица 4 – таблица «Workers».

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| WorkerID | PRIMARY KEY, INTEGER, NOT NULL |
| Name | NVARCHAR(50), NOT NULL |
| LastName | NVARCHAR(100), NOT NULL |
| PhoneNamber | NVARCHAR(20), NOT NULL |
| Position | NVARCHAR(20), NOT NULL |
| Experience | NVARCHAR(20), NOT NULL |
| ForemenId | FOREIGN KEY, INTEGER, NULL |

1. Проектирование работы системы

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

При проектировании работы системы строительной фирмы 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 заказчиков, прорабов, рабочих и объектах 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 заказчиков, прорабов, рабочих и об объектах. В рамках создания CRUD функционала важно предусмотреть возможности добавления, просмотра, обновления и удаления записей.

Для заказчика системы необходимо предусмотреть функционал регистрации в системе, добавление заявки на строительства объекта. 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 перечень заказов на строительство, перечень строительных объектов у выбранной бригада находящихся в стадии выполнения, перечень завершенных объектов, выручка от выполненных работ в целом, выбранной бригадой за любой период времени.

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 строительной фирмы включает в себя проектирование базы данных, разработку пользовательского интерфейса, создание CRUD операций для администратора, заказчика. 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 строительной фирмы.

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

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

Модель «Заказчик» представляет собой класс для хранения информации о заказчике строительной фирмы. О заказчике должна храниться следующая информация: Имя, Фамилия и номер телефона. Нужно учитывать, что информация о заказчике будет использована и в модели для объекта, т.е. будет создан внешний ключ, связывающий объект и заказчик. Для модели была продумана следующая структура класса:

- CustomerID – Уникальный идентификатор заказчика;

- Name – Имя заказчика;

- LastName – Фамилия заказчика;

- PhoneNamber – Номер телефона заказчика.

Реализация модели:

Листинг кода 1 – Класс для реализации модели Заказчик Customer.cs

public class Customer

{

[Key]

public int CustomerID { get; set; }

[Display(Name = "Имя"), Required, MaxLength(50)]

public string Name { get; set; }

[Display(Name = "Фамилия"), Required, MaxLength(100)]

public string LastName { get; set; }

[Display(Name = "Номер телефона"), Required, MaxLength(20)]

public string PhoneNamber { get; set; }

}

Модель «Объект» представляет собой класс для хранения информации об объекте строительной фирмы. Об объекте необходима следующая информация: Название, Адрес, Тип, Статус и фотография. Как упоминалось выше, должна храниться информация о заказчике объекта (заказчик строительной фирмы) и прорабе, в связи с чем необходимо связать эти модели с помощью поля класса и создания внешнего ключа, связывающего объект и заказчика соответственно.

Для модели была продумана следующая структура класса:

- OobjectID – Уникальный идентификатор объекта;

- Title – Название объекта;

- Adress – Адрес объекта;

- Type – Тип объекта;

- Status – Статус объекта;

- Photo – Фотография объекта;

- ForemenId – Идентификатор прораба;

- Foremen – экземпляр класса Foremen, соответствующий идентификатору прораба (ForemenId).

- CustomerId – Идентификатор заказчика;

- Customer – экземпляр класса Customer, соответствующий идентификатору заказчика (CustomerId).

Реализация модели:

Листинг кода 2 – Класс для реализации модели Объект Oobject.cs:

public class Oobject

{

[Key]

public int OobjectID { get; set; }

[Display(Name = "Название"), Required, MaxLength(50)]

public string Title { get; set; }

[Display(Name = "Адрес"), Required, MaxLength(130)]

public string Adress { get; set; }

[Display(Name = "Тип"), Required, MaxLength(50)]

public string Type { get; set; }

[Display(Name = "Статус"), Required, MaxLength(50)]

public string Status { get; set; }

[Display(Name = "Фото")]

public string? Photo { get; set; }

public int? ForemenId { get; set; }

public Foremen? Foremen { get; set; }

public int? CustomerId { get; set; }

public Customer? Customer { get; set; }

}

Модель «Прораб» представляет собой класс для хранения информации о прорабе строительной фирмы. О прорабе должна храниться следующая информация: Имя, Фамилия, Квалификация, Специализация, Навыки и Номер телефона. Нужно учитывать, что информация о прорабе будет использована и в модели объект, т.е. будет создан внешний ключ, связывающий прораба и объекта. Для модели была продумана следующая структура класса:

- ForemenID – Уникальный идентификатор прораба;

- Name – Имя прораба;

- LastName – Фамилия прораба;

- PhoneNamber – Номер телефона прораба;

- Qualification – Квалификация проживания прораба;

- Specialization – Специализация проживания прораба;

- Skills – Навыки прораба;

Реализация модели:

Листинг кода 3 – Класс для реализации модели Прораб ForemenID.cs:

public class Foremen

{

[Key]

public int ForemenID { get; set; }

[Display(Name = "Имя"), Required, MaxLength(50)]

public string Name { get; set; }

[Display(Name = "Фамилия"), Required, MaxLength(100)]

public string LastName { get; set; }

[Display(Name = "Номер телефона"), Required, MaxLength(100)]

public string PhoneNamber { get; set; }

[Display(Name = "Квалификация"), Required, MaxLength(100)]

public string Qualification { get; set; }

[Display(Name = "Специализация "), Required, MaxLength(100)]

public string Specialization { get; set; }

[Display(Name = "Навыки"), Required, MaxLength(20)]

public string Skills { get; set; }

}

Модель «Рабочий» представляет собой класс для хранения информации о рабочим строительной фирмы. О рабочим должна храниться следующая информация: Имя, Фамилия, Номер телефона, Должность и Опыт. Нужно учитывать, что информация об прорабе будет использована и в модели рабочий, т.е. будет создан внешний ключ, связывающий рабочий и прорабе. Для модели была продумана следующая структура класса:

- WorkerID – Уникальный идентификатор рабочего;

- Name – Имя рабочего;

- LastName – Фамилия рабочего;

- PhoneNamber – Номер телефона рабочего;

- Position – Должность рабочего;

- Experience – Опыт рабочего.

- ForemenId – Идентификатор прораба;

- Foremen – экземпляр класса Foremen, соответствующий идентификатору прораба (ForemenId).

Реализация модели:

Листинг кода 4 – Класс для реализации модели Рабочий Worker.cs:

public class Worker

{

[Key]

public int WorkerID { get; set; }

[Display(Name = "Имя"), Required, MaxLength(50)]

public string Name { get; set; }

[Display(Name = "Фамилия"), Required, MaxLength(100)]

public string LastName { get; set; }

[Display(Name = "Номер телефона"), Required, MaxLength(20)]

public string PhoneNamber { get; set; }

[Display(Name = "Должность"), Required, MaxLength(20)]

public string Position { get; set; }

[Display(Name = "Опыт"), Required, MaxLength(20)]

public string Experience { get; set; }

public int? ForemenId { get; set; }

public Foremen? Foremen { 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 FirstName, LastName, Email, Password и ConfirmPassword.

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: имени и фамилии. Также были перенастроена web-страница регистрации.

Листинг кода 6 – Класс для реализации модели пользователя:

public class ConstructionUser : IdentityUser

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

Листинг кода 7 – Изменения в файле Register.cshtml:

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

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

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

aria-required="true" placeholder="First name" />

<label asp-for="Input.FirstName">Имя:</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" autocomplete="username"

aria-required="true" placeholder="Last name" />

<label asp-for="Input.LastName">Фамилия:</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 строительной фирмы. 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 строительной фирмы с использованием технологии ASP.NET Core MVC, созданы концептуальная, логическая и физическая модели данных. В результате работы создано приложение для хранения информации объектов строительной фирмы, заказчиков, прорабов, рабочих 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

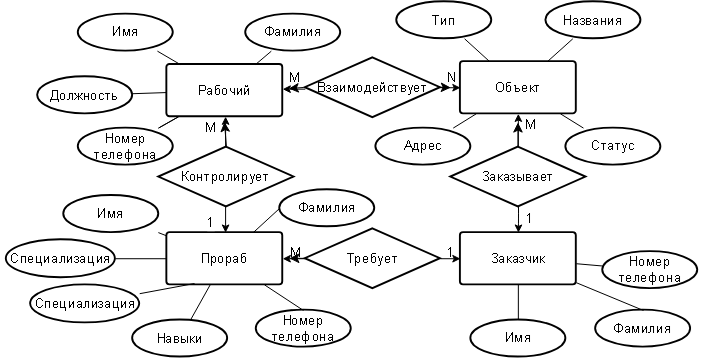


Рисунок 1 – Концептуальная модель данных

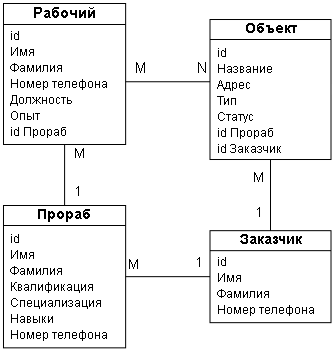


Рисунок 2 – Логическая модель данных

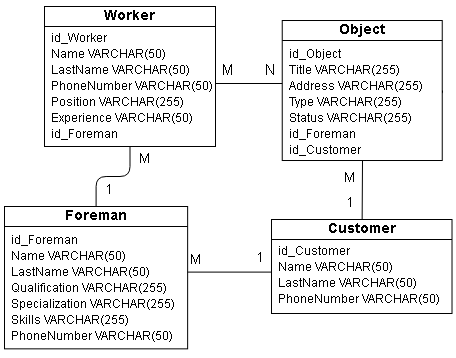


Рисунок 3 – Физическая модель данных

Приложение 2

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

Приложение 3

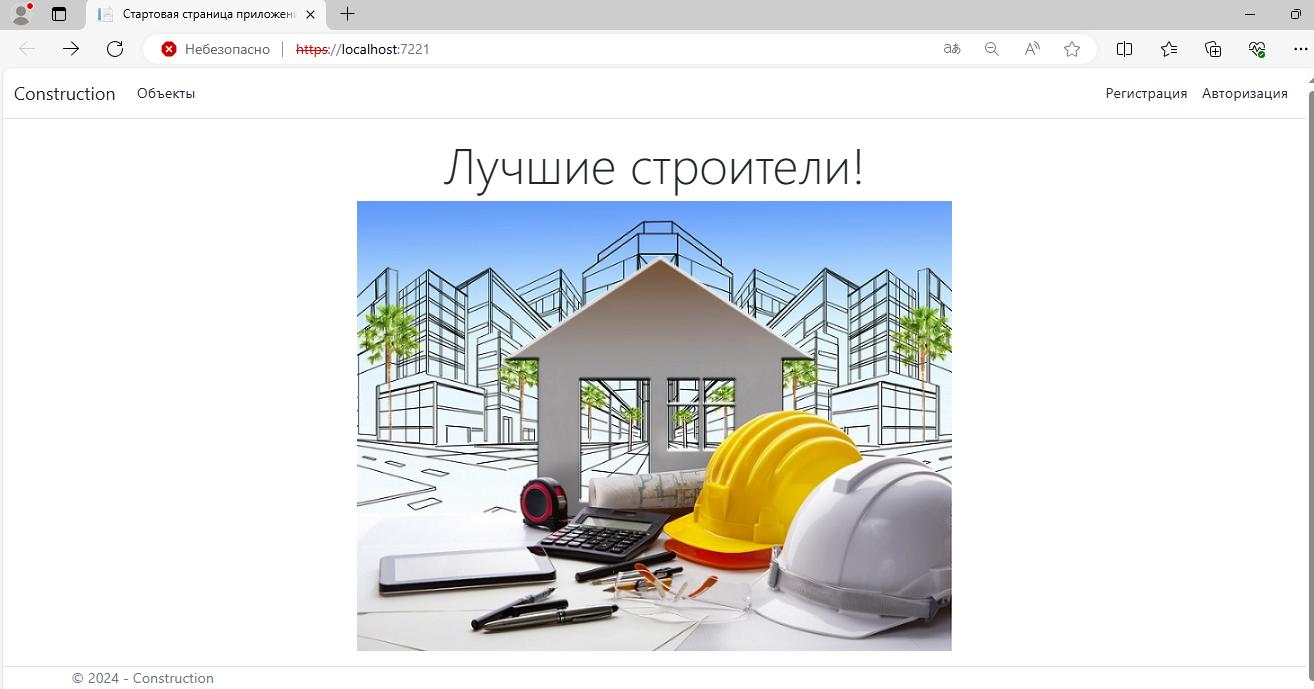


Рисунок 1 – Стартовая страница приложения

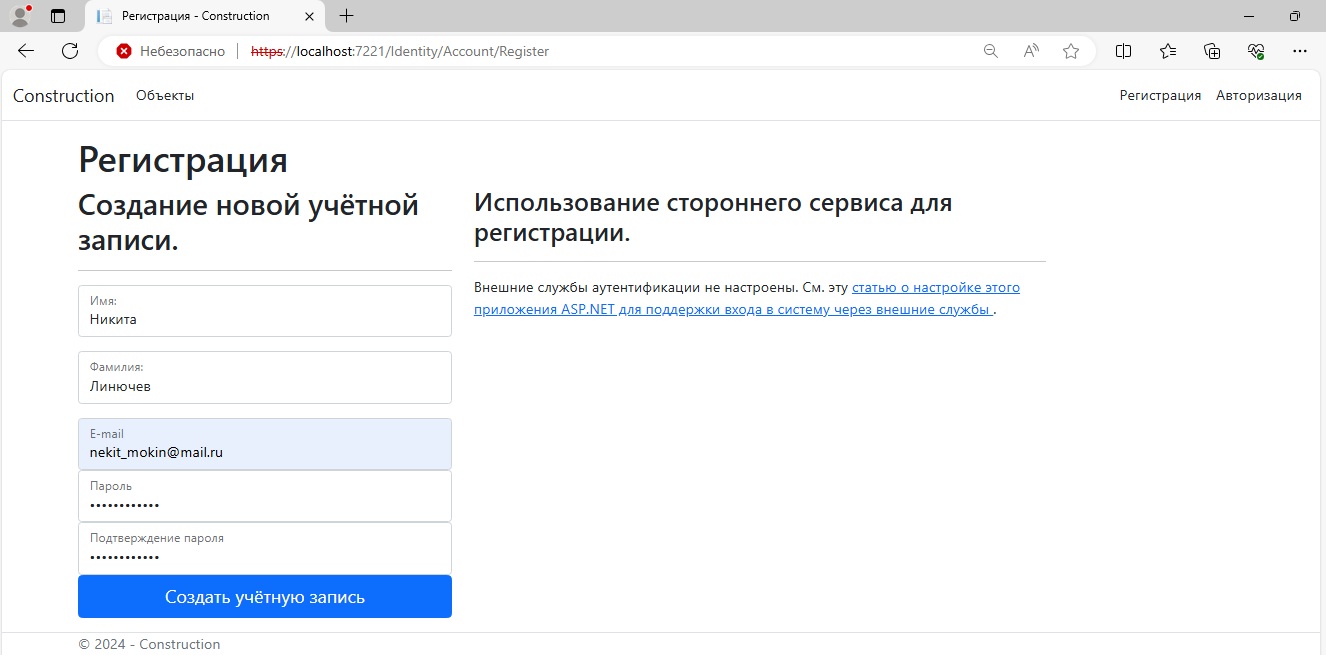


Рисунок 2 – Страница регистрации

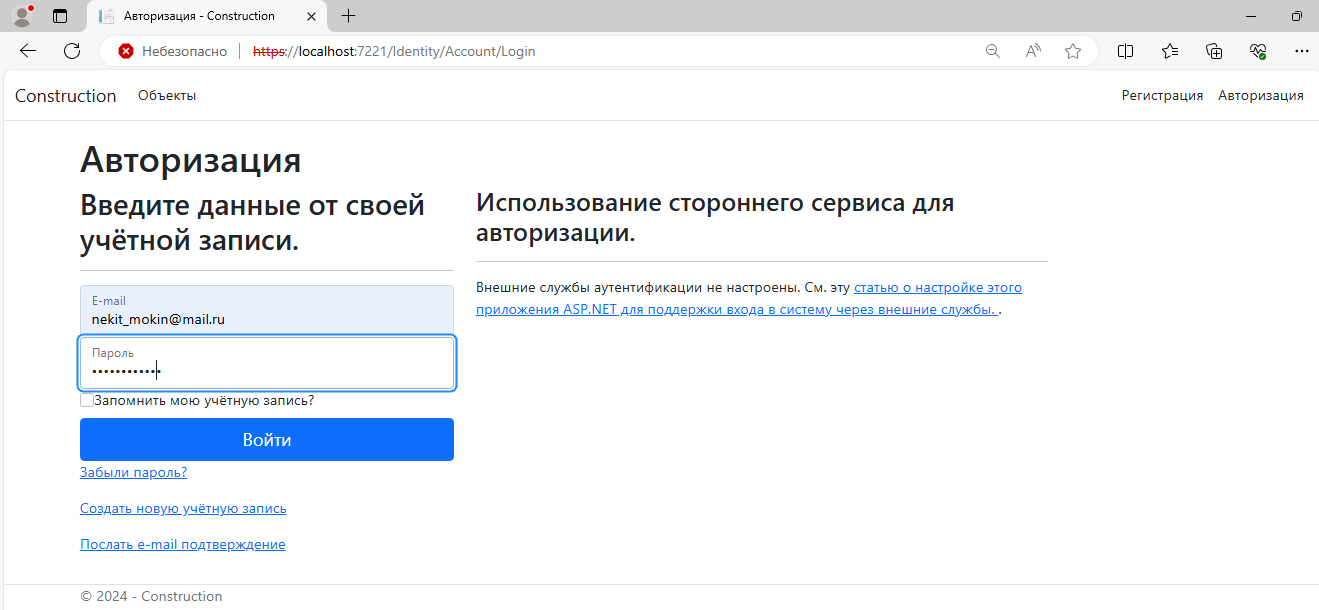


Рисунок 3 – Страница авторизации

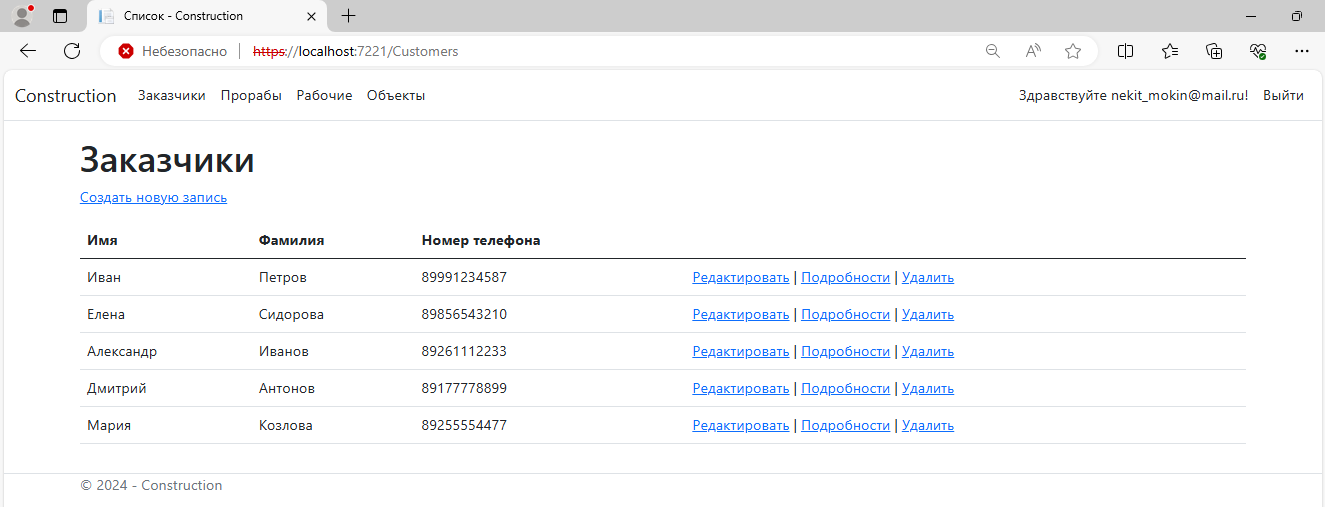


Рисунок 4 – Список заказчиков

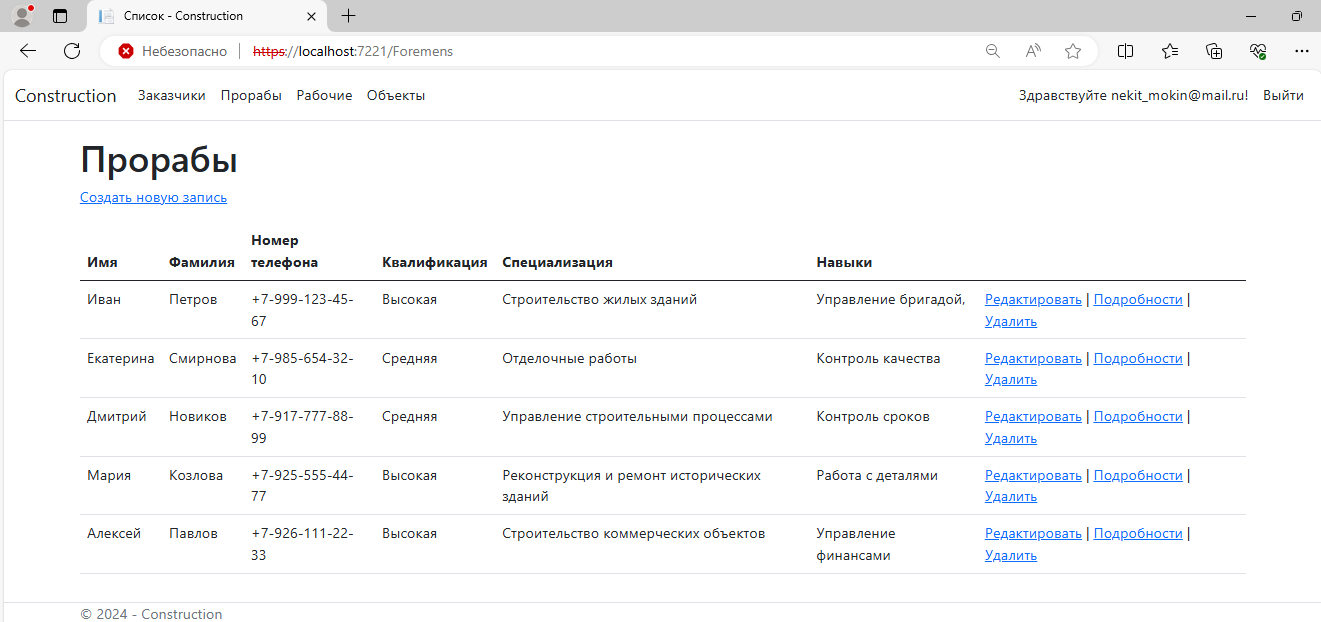


Рисунок 5 – Список прорабов

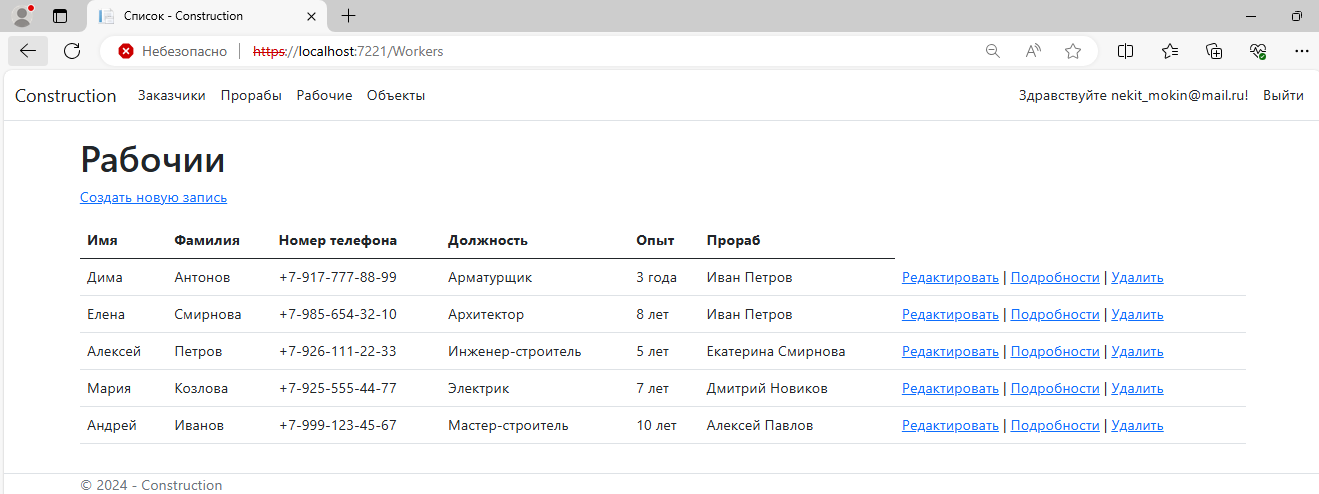


Рисунок 6 – Список рабочих

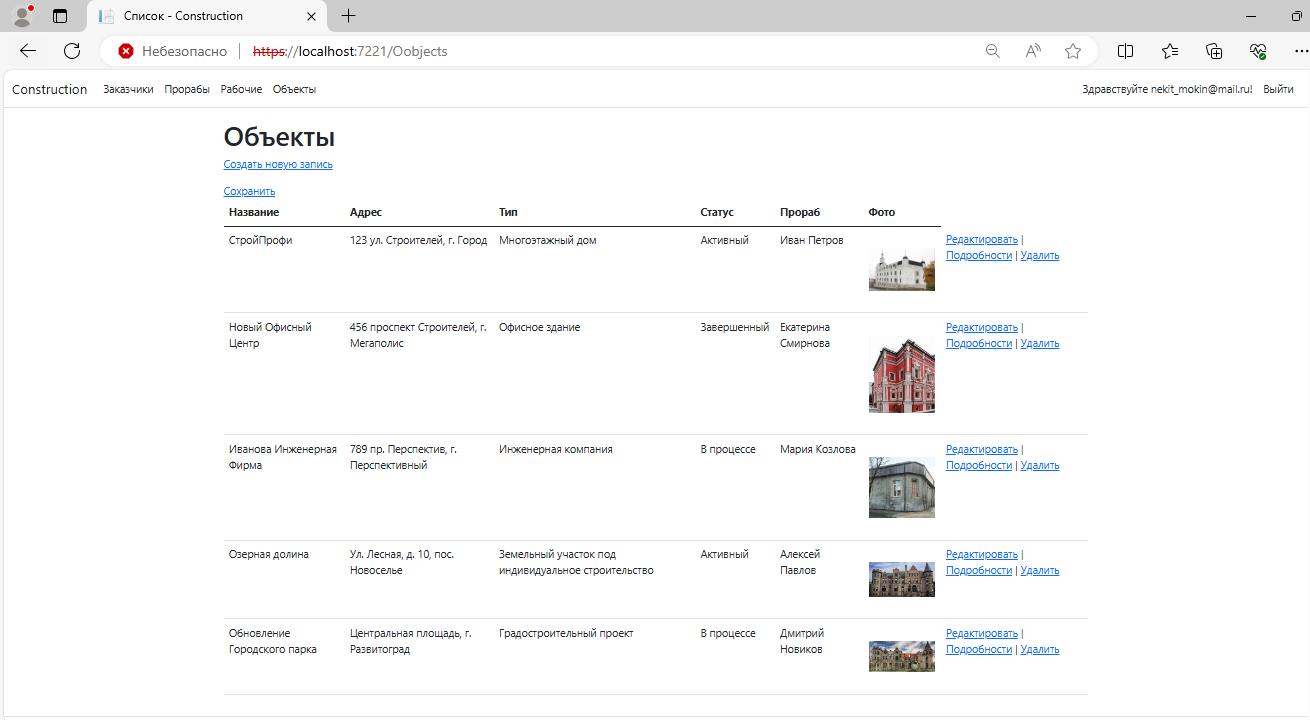


Рисунок 7 – Список объектов

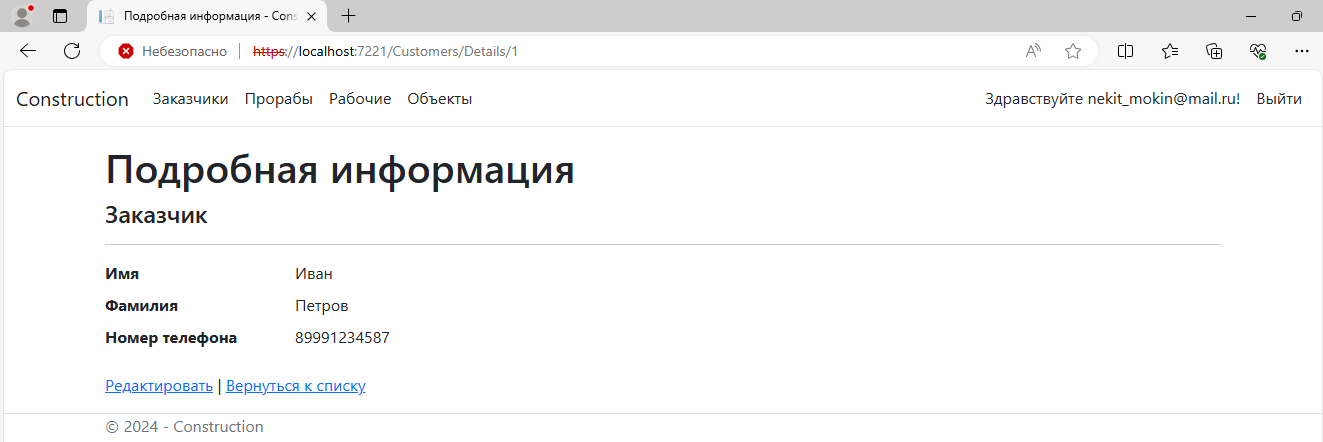


Рисунок 8 – Страница подробной информации о заказчике

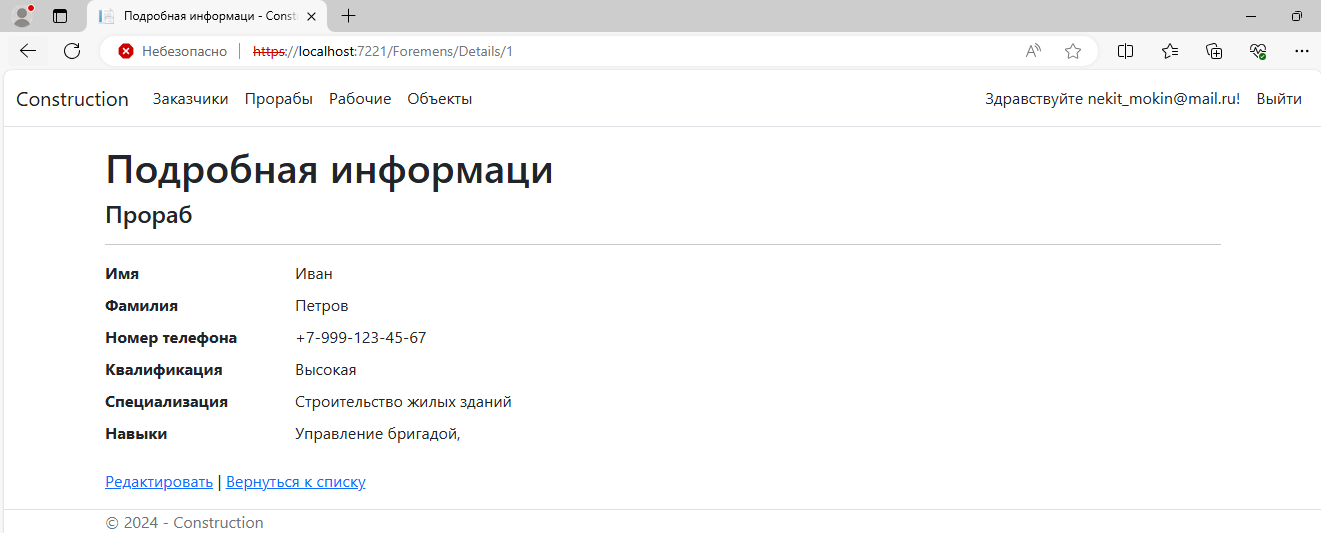


Рисунок 9 – Страница подробной информации о прорабе

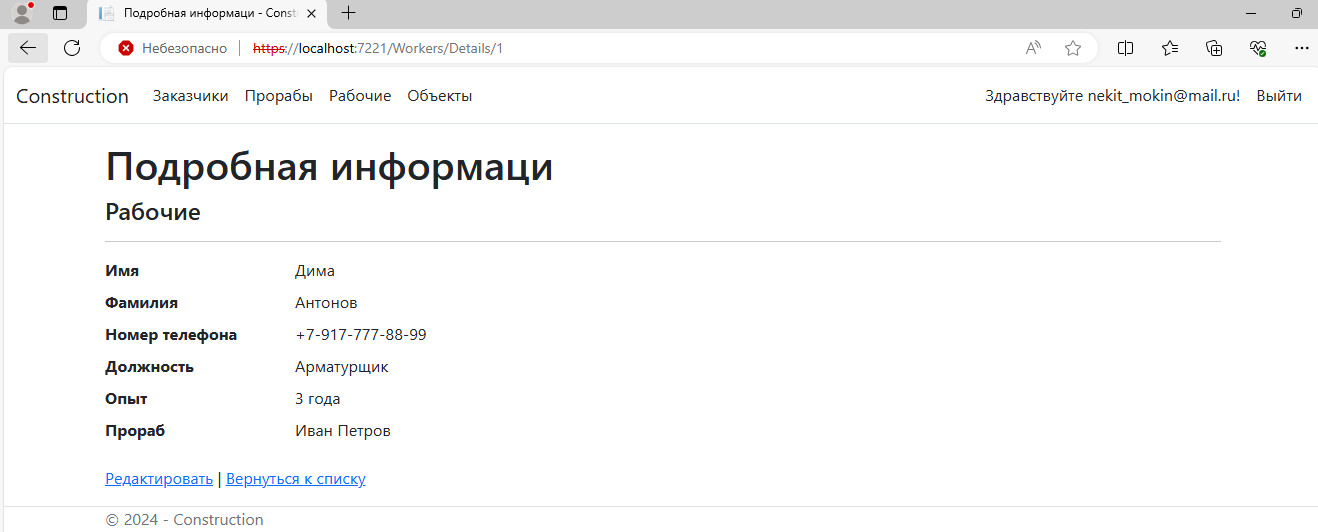


Рисунок 10 – Страница подробной информации о рабочем

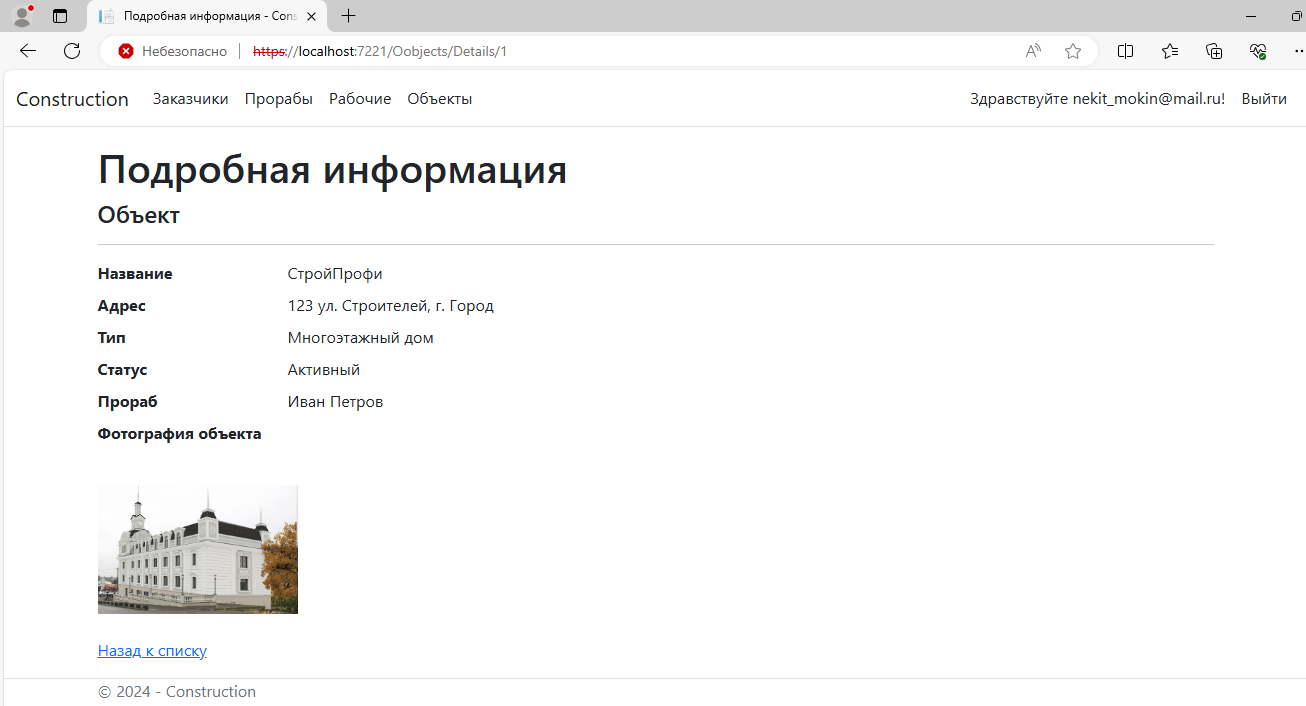


Рисунок 11 – Страница подробной информации об объекте

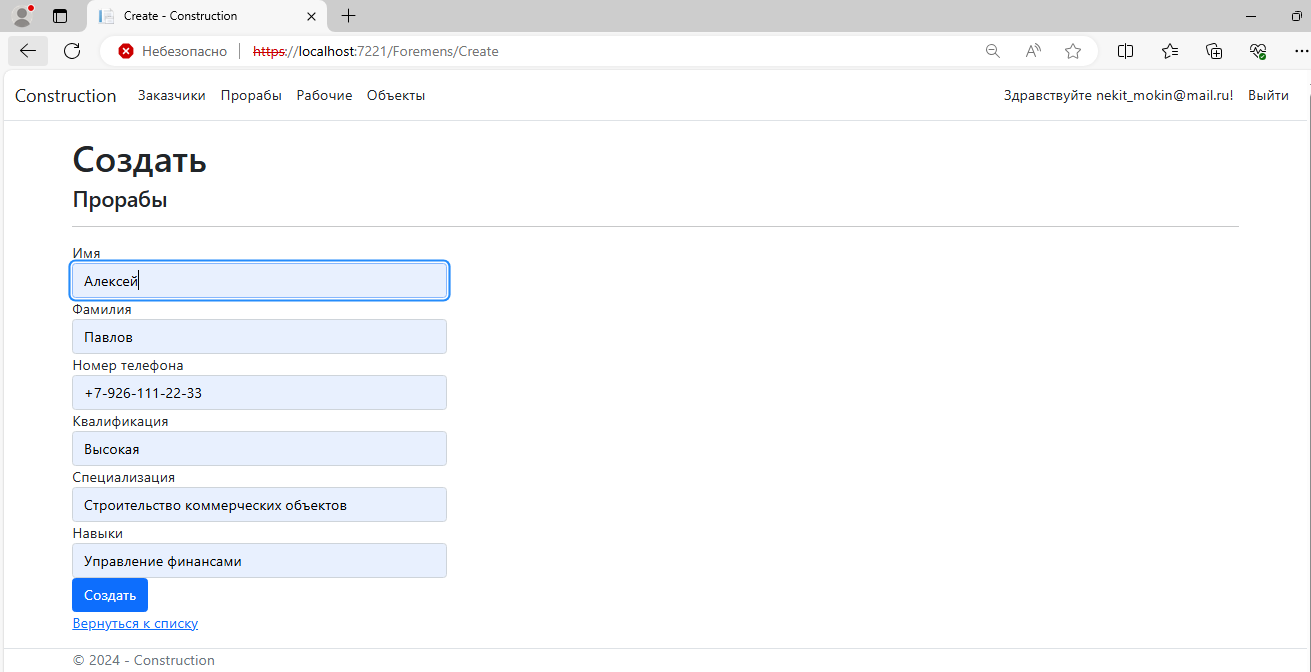


Рисунок 12 – Страница добавления прораба

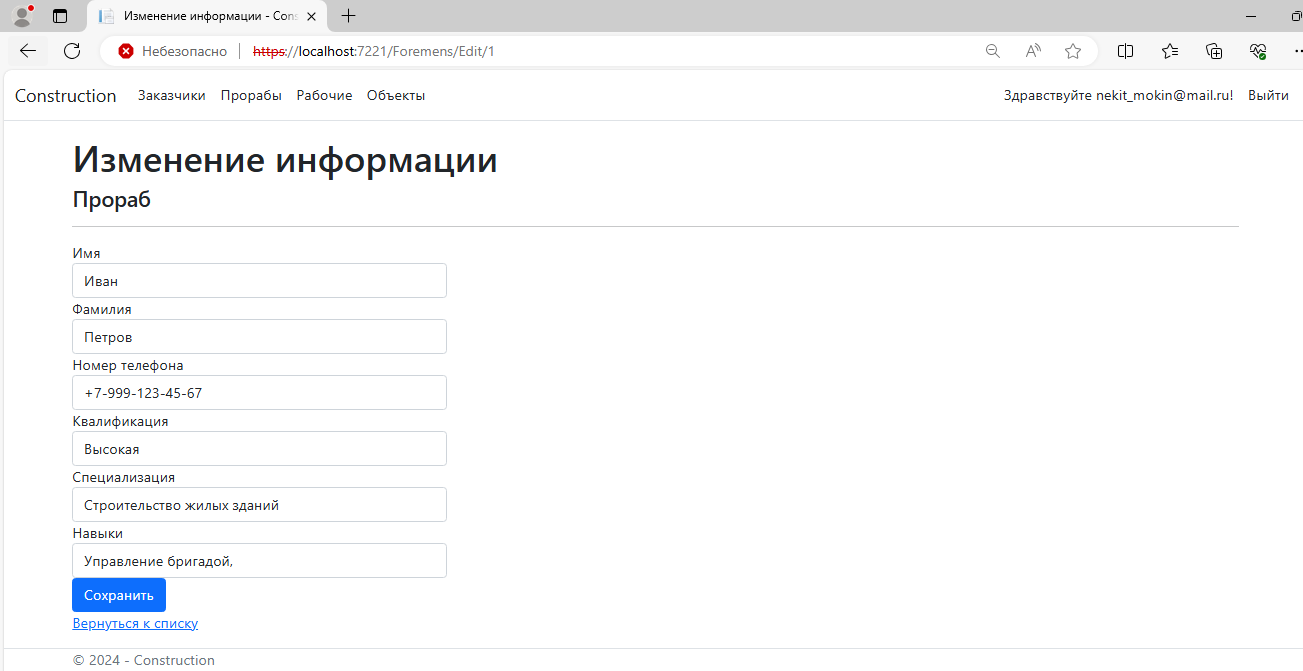


Рисунок 13 – Страница изменения информации о прорабе

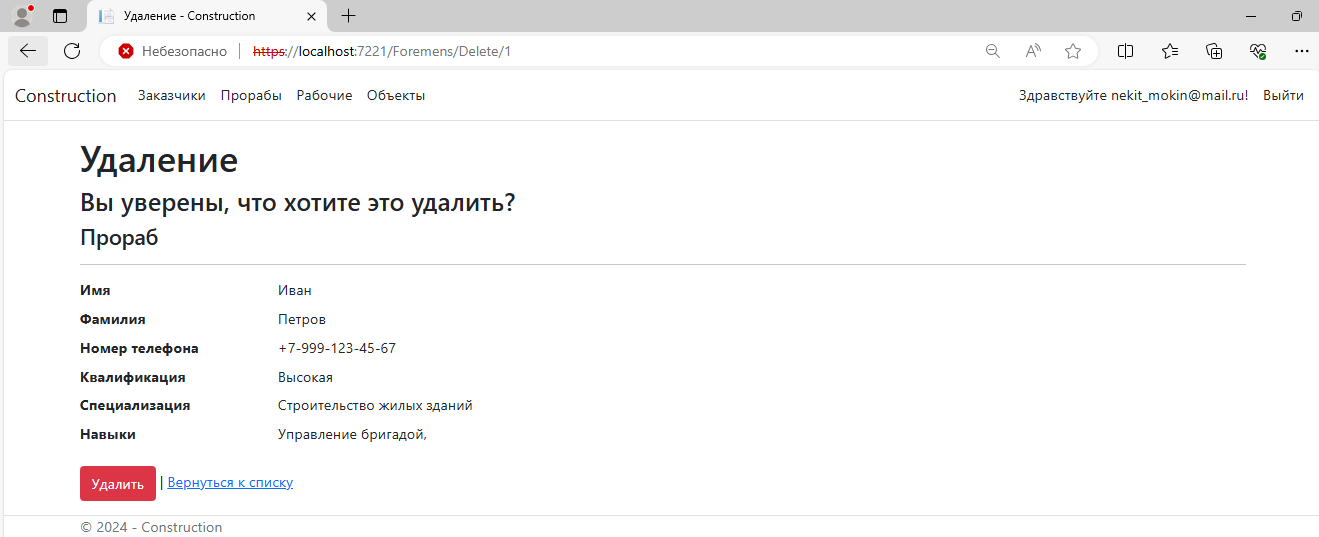


Рисунок 14 – Страница удаления прораба

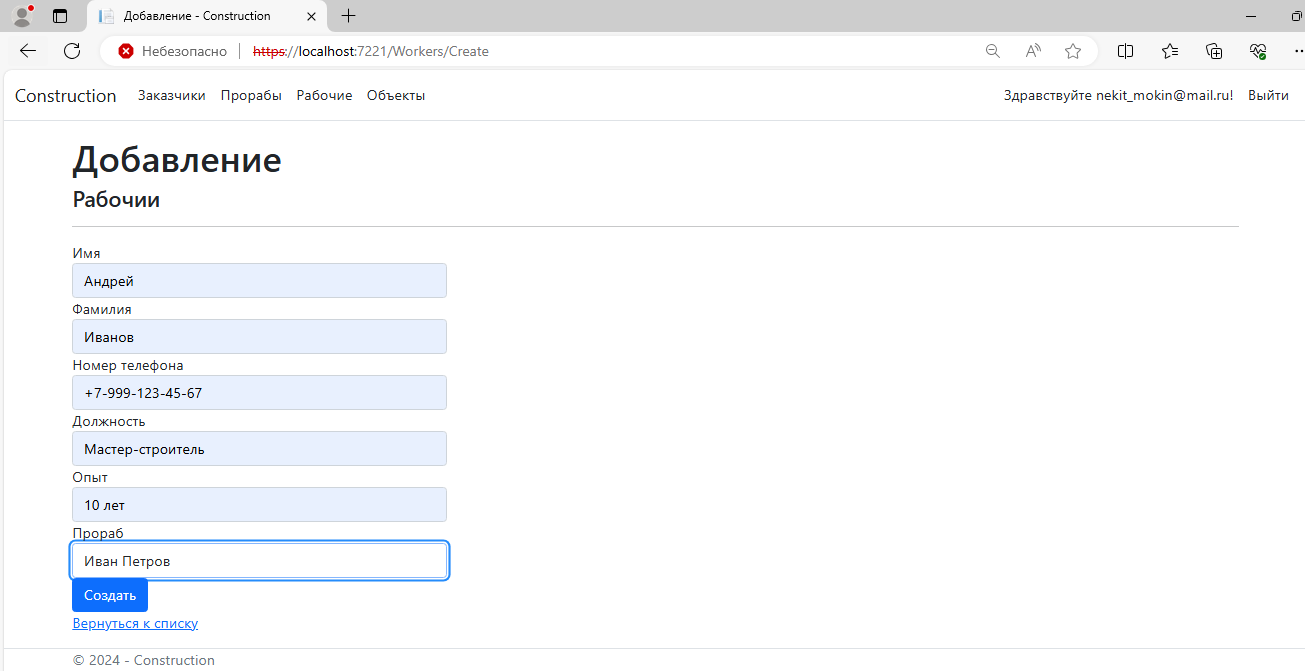


Рисунок 15 – Страница добавления рабочего

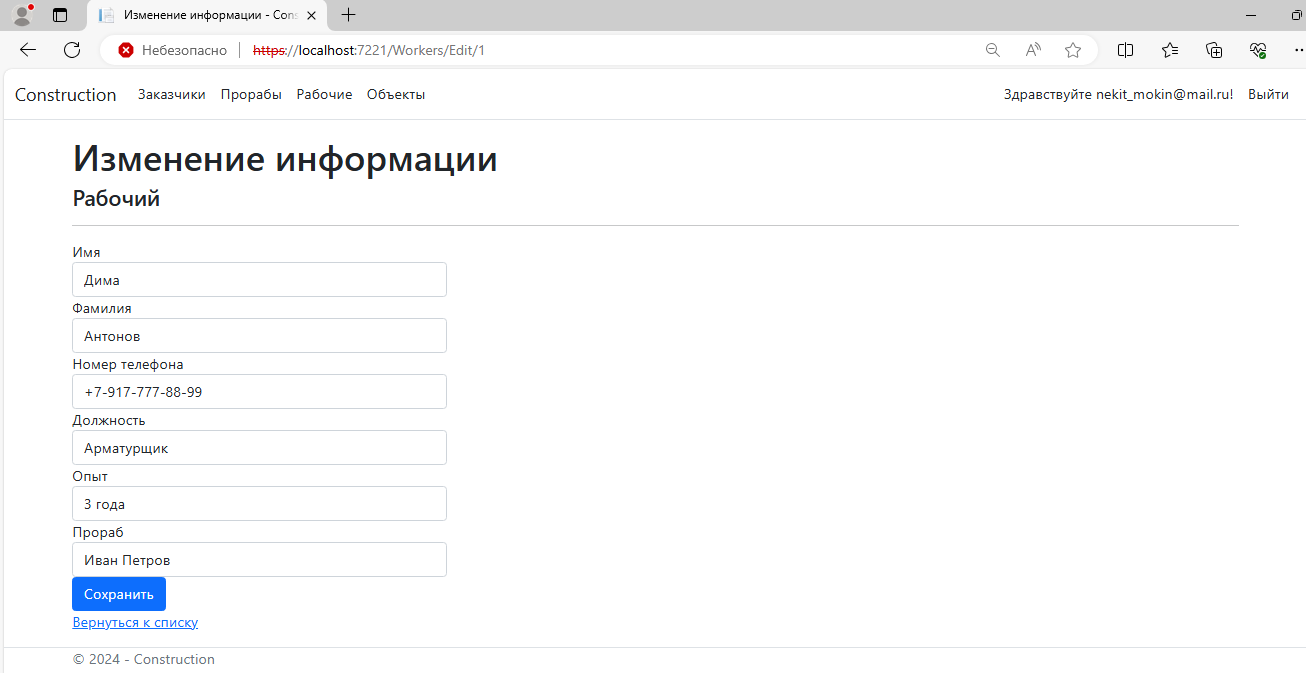


Рисунок 16 – Страница изменения информации рабочего

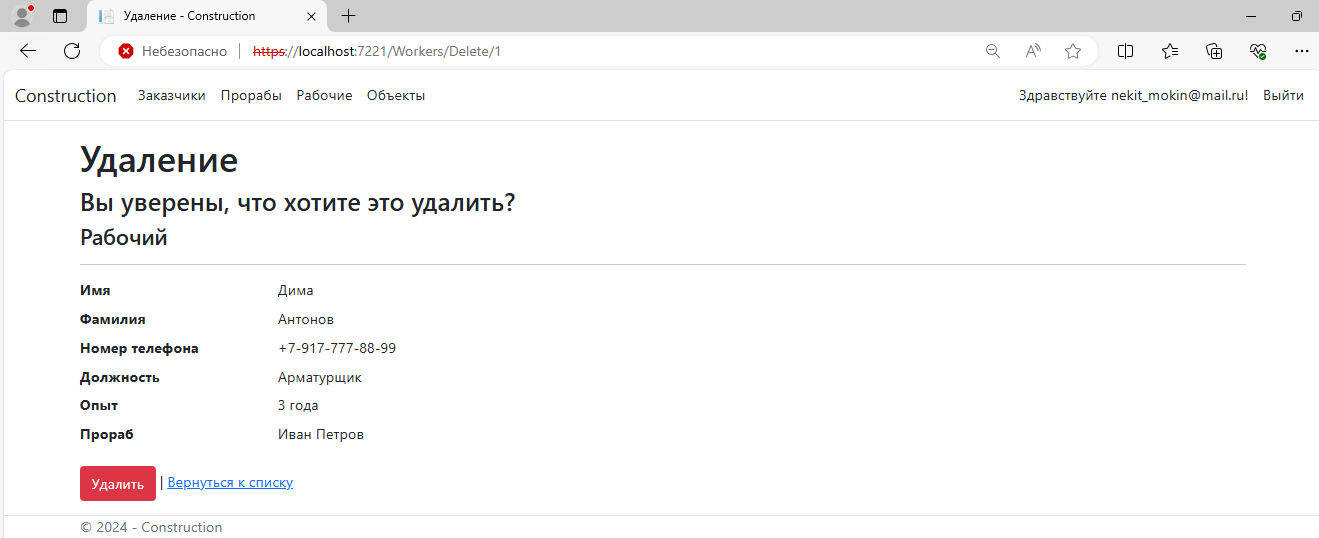


Рисунок 17 – Страница удаления рабочего

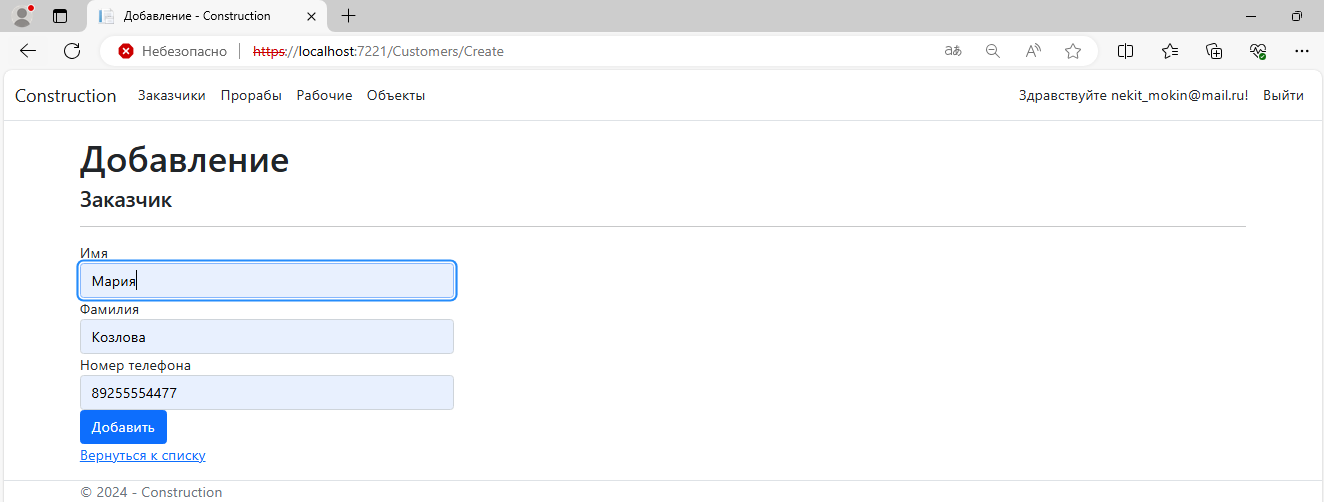


Рисунок 18 – Страница добавления заказчика

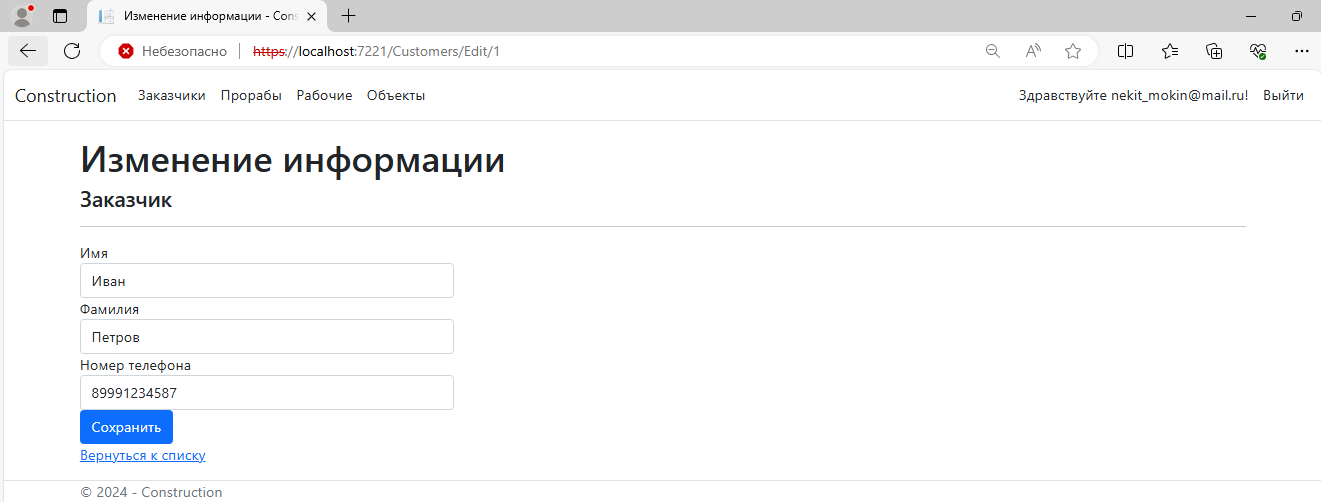


Рисунок 19 – Странице изменения информации заказчика

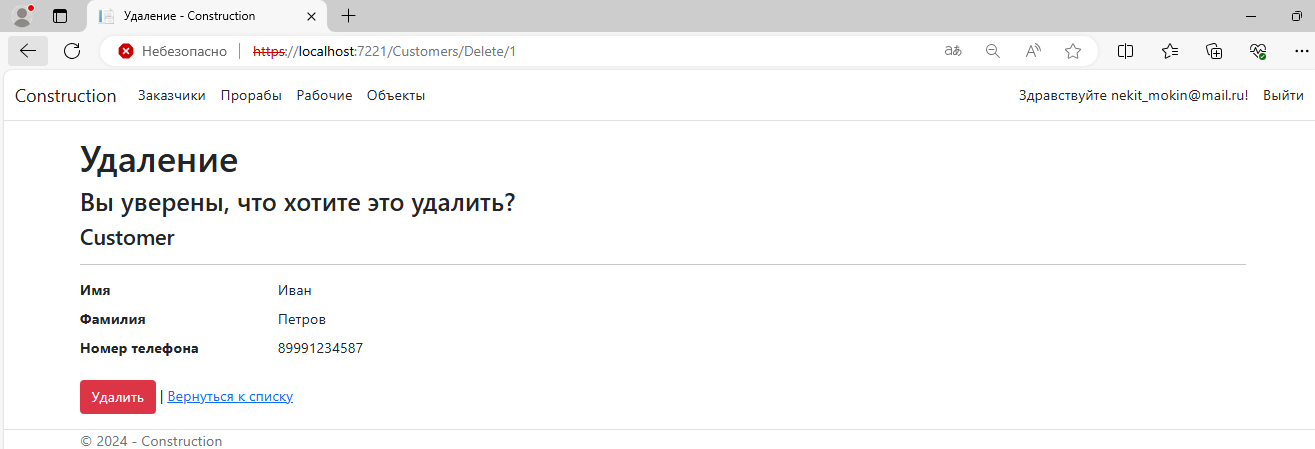


Рисунок 20 – Страница удаления заказчика

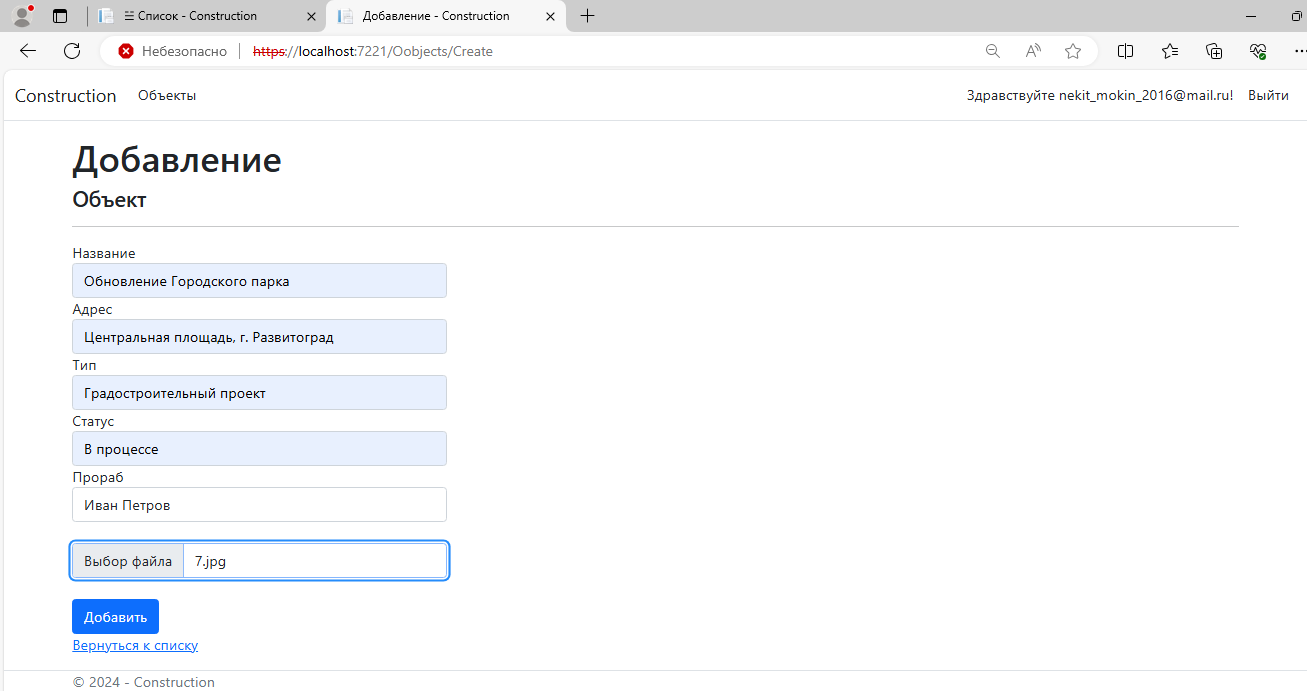


Рисунок 21 – Страница добавления объекта со стороны заказчика

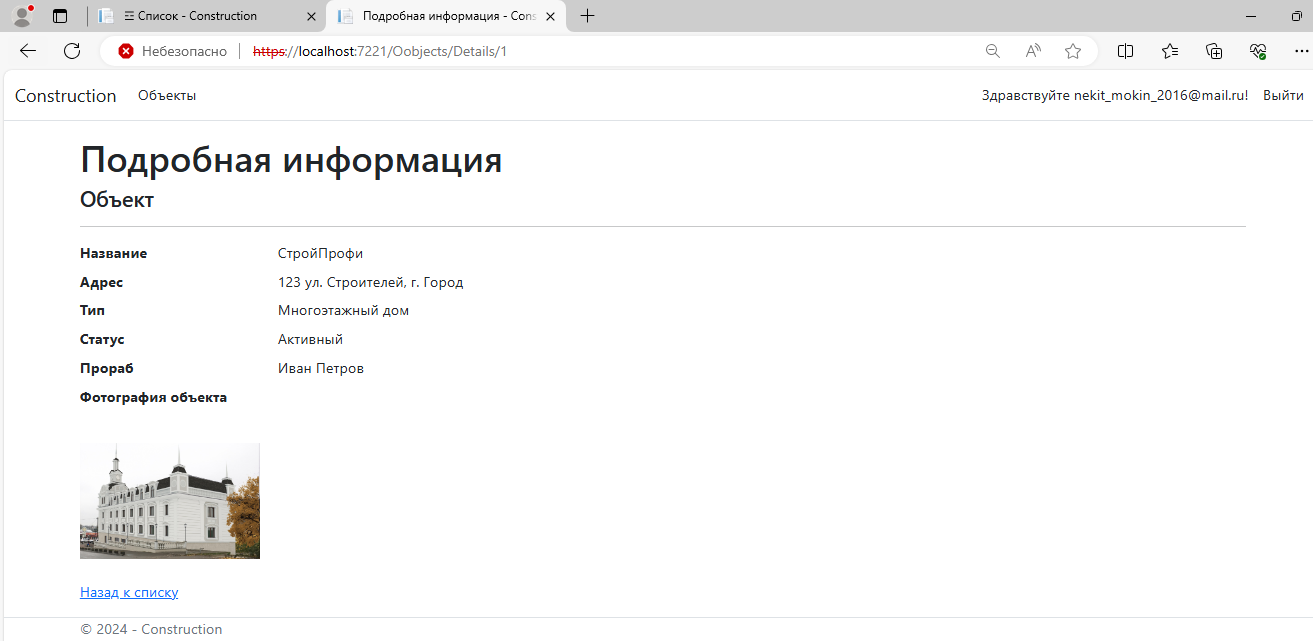


Рисунок 22 – Страница подробной информации об объекте

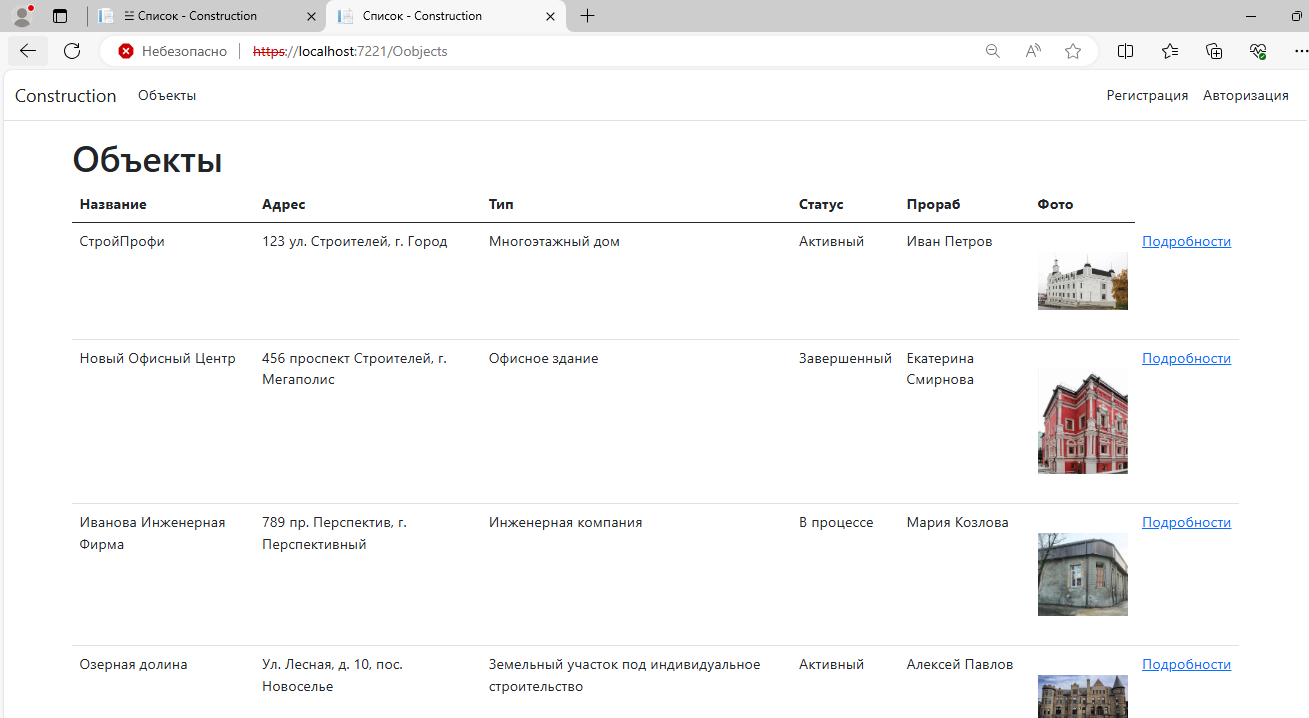


Рисунок 23 – Проверка отображение объектов для незарегистрированных пользователей

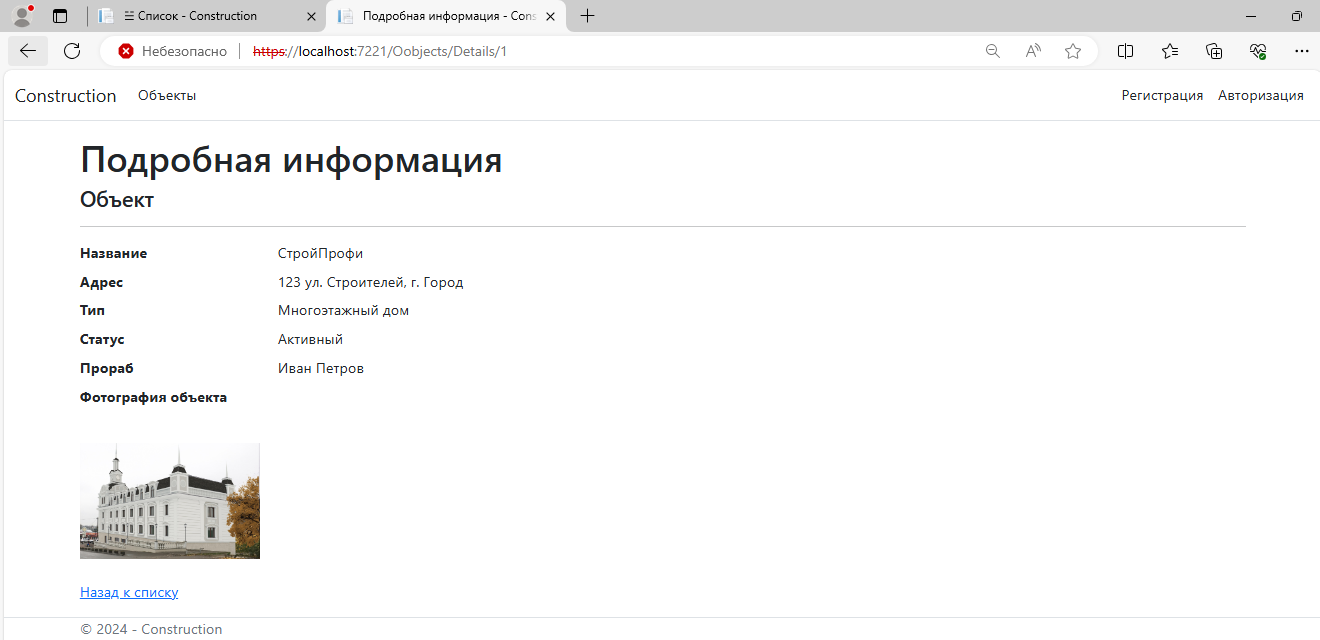


Рисунок 24– Проверка отображение подробной информации объектов для незарегистрированных пользователей

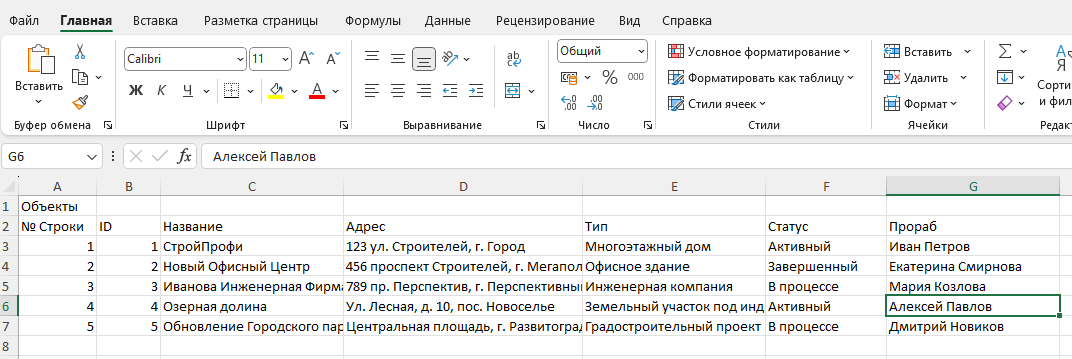


Рисунок 25 – Полученный отчёт «Список объектов»