МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет *информационных систем и технологий*

Кафедра *Измерительно-вычислительные комплексы*

Дисциплина *Базы данных*

КУРСОВАЯ РАБОТА

Тема *«Автоматизированная информационная система бюро экспертизы и оценки»*

Выполнил студент _____ /*А.А. Ульянин*/

подпись *инициалы, фамилия*

Курс *2* Группа *ИСТбд-21*

Направление *09.03.02 «Информационные системы и технологии»*

Руководитель *доцент кафедры ИВК, к.т.н., доцент*

должность, учёная степень, учёное звание

Родионов Виктор Викторович

фамилия,имя, отчество

Дата сдачи:

«___» _____ 20___ г.

Дата защиты:

«___» _____ 20___ г.

Оценка:_____

Ульяновск

2021

ИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет *информационных систем и технологий*

Кафедра *Измерительно-вычислительные комплексы*

Дисциплина *Базы данных*

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

студенту *ИСТбд - 21* *Ульянин А.А.*
группа *фамилия, инициалы*

Тема работы *«Автоматизированная информационная система бюро экспертизы и оценки»*

Срок сдачи законченной работы «___» _____ 20___ г.

Исходные данные к работе *методические указания к выполнению курсовой работы и проведению практических занятий для студентов направления 09.03.03 «Информационные системы и технологии» по дисциплине «Базы данных» Родионов В.В.*
(базовое предприятие, характер курсовой работы: задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

Содержание пояснительной записки *введение, техническое задание, информационное обеспечение системы, алгоритмическое обеспечение системы, прикладное программное обеспечение системы, руководство пользователя, заключение, список использованных источников.*

Перечень графического материла _____

Руководитель *доцент каф. ИВК* _____ */ В.В. Родионов /*
должность *подпись* *инициалы, фамилия*

«___» _____ 20___ г.

Студент _____ */ А. А. Ульянин /*
подпись *инициалы, фамилия*

«___» _____ 20___ г.

Содержание

Введение

В ходе выполнения данной работы я подробно изучил тему: «Бюро экспертизы и оценки». Бюро Экспертизы и оценки - специализированная организация, осуществляющая в разных областях экспертизу с последующей оценкой, результат которой может стать решающим в вынесении вердикта по спорному вопросу. Такую экспертизу проводят профессионалы, обладающие знаниями в своих предметных областях, а производят её с помощью комплекта профессионального оборудования. Заключение специалиста помогает установить истину, оценить причиненный ущерб, попытаться описать решение по объекту экспертизы, выявить стоимость объекта.

«Бюро Экспертизы и Оценки» осуществляет проведение качественно выполненных экспертных исследований в обширном спектре. Бюро проводит экспертные исследования по следующим направлениям: проведение судебных экспертиз по требованию следственного комитета, внесудебные экспертизы по заявлениям частных лиц , рецензирование всех типов заключений , консультирование различных лиц по любым вопросам, связанные с назначением экспертизы .

Бюро в зависимости от квалификации сотрудников и наличия лицензии может осуществлять работу в различных областях. Это может быть агротехническая, строительная, пожарно-техническая, техническая, экологическая, видеофонографическая, фоноскопическая, лингвистическая, почерковедческая, психологическая, товароведческая, экономическая, юридическая и другие виды экспертизы. Также они проводят оценку недвижимости, транспорта, бизнеса, оборудования различной специализации.

Составляющие экспертизы: субъект, объект, критерии, методы, процедура и результат. Субъект экспертизы - это эксперт или группа экспертов. Объектом экспертизы являются потребительские свойства товаров, проявляющиеся при взаимодействии товара с потребителем в процессе эксплуатации. Критерии, используемые в экспертной оценке, могут быть общими и конкретными. Общие

критерии - это сложившиеся в обществе ценностные представления, ориентации и нормы. Конкретные критерии - реальные требования к качеству товаров данного вида. Эти требования определены нормативно-технической документацией. Конкретными критериями могут служить также базовые образцы и базовые показатели, характеризующие качество образцов, принимаемых за исходные.

В товарной экспертизе используются разнообразные методы экспертизы: физико-технические, химические, биологические, математические и др., исследования проводятся с применением сложных современных приборов и технических средств.

Основные операции процедуры экспертизы можно разделить на три этапа: подготовительный (создание экспертной группы и формирование целей экспертизы); основной (исследования, выполняемые экспертами); заключительный (обработка результатов, их анализ, оценка и оформление экспертного заключения).

Результатом экспертизы является оформление в письменном виде заключения, в котором приводится оценка потребительских свойств. Заключение включает вводную часть, исследовательскую часть и вывод; подписывает ее эксперт.

История бюро корнями уходит в глубокое прошлое. Слово «эксперт» в переводе с французского означает «знающий». Первая экспертная организация занималась исследованием почерков и подписей. Она была открыта в 1595 году в Париже. Впервые это понятие возникло, когда были необходимы официальные исследования и заключения, подтверждаемые специалистами.

Бюро актуально и по сей день. У каждого человека есть своя недвижимость и техника. Со временем возникают некие ситуации: поломка техники, причинённый ущерб техники, перепродажа недвижимости. Чтобы дать оценку и возможное решение проблеме, как раз и существует данное бюро.

Её важность заключается в защите своих прав и своих собственных интересов: желание продать объект экспертизы по дороже, прибегнуть к обмену и т.п.

Моя система специализируется на экспертизе и оценке, представляет собой обобщённую модель бюро по проведению экспертизы с последующей оценкой техники и недвижимости. Для того, чтобы создать ее я использовал данные источники:

1. Электронные ресурсы: «Ростовский центр судебных экспертиз», «Судебный Эксперт» - судебная и внесудебная независимая экспертиза: «Список проводимых экспертиз», «Областное Бюро Экспертизы и Оценки» - помогли мне создать представление о необходимых данных для создания базы данных и её аргументов по моей тематике.

2. Организация и проведение экспертизы и оценки качества товаров Книга помогла понять, что такое экспертиза и экспертная оценка, её задачи, а также, представление о товарной экспертизе.

3. «Областное Бюро Экспертизы и Оценки» оказала мне также помощь в оформлении «Введения». Предоставил мне основные цели бюро проведения экспертизы.

4. «AG – полезная информация для вас» - помог мне понять, какие существуют задачи у бюро экспертизы и оценки. А также оказал помощь в оформлении введения. А именно предоставил мне понятие бюро экспертизы, а также её историю создания.

5. Руководство по ASP.NET MVC 5 был полезен для меня в ходе создания контроллеров, фильтрации .

6. Документация по ASP.NET MVC// Майкрософт был полезен при создании программной части своей системы.

7. ГОСТ Р.7.0.100-2018 Система стандартов по информации, библиотечному и издательскому делу - предоставил мне основные данные по оформлению библиографического списка.

1. Техническое задание

1.1 Общие сведения

Автоматизированная информационная система, далее система «Бюро экспертизы и оценки».

1. 2 Назначение и цели создания системы

1. 2. 1 Назначение системы

Область автоматизируемой деятельности рассчитана на проведение экспертизы с последующим оцениванием недвижимости и техники. Система предназначена для организации бюро экспертизы и оценки, в обеспечении удобства при обращении клиентов, а также для просмотра и редактирования информации.

В системе содержаться личные данные клиентов, поэтому она предназначена только для пользования самой организацией.

1. 2. 2 Цели создания системы

1) Цель заключается в проведении точной оценки объектов экспертизы от конкретного формирования целей и содержания до определения результата.

2) Цель заключается в предоставлении удобного мониторинга накладных с упрощённым слежением за процессом «экспертизы и оценки» с быстрым доступом объектов экспертизы.

1.3 Характеристика объекта автоматизации

Бюро Экспертизы и Оценки предлагает перечень услуг по экспертизе с последующей оценкой в сфере недвижимости и техники, индивидуальный подход к каждому клиент, надежность, сжатые сроки, доступные цены при высоком качестве.

Бюро помогает разрешить спорные ситуации и проводит независимую экспертизу/оценку на должном уровне.

Сотрудники являются экспертами своего дела, они обладают обширными знаниями в своих предметных областях.

Производятся экспертизы недвижимости на качество ремонта, даётся экс-

пертное заключение по определению стоимости ущерба, нанесённого в результате залива, пожара и иных аварийных ситуаций, проводятся экспертизы смет (соответствие их проекту, стоимости, соразмерности цен) и предлагается решение по устранению той или иной проблемы. Ряд мероприятий строительной экспертизы направлен на выявление возможных нарушений подрядчика, что важно для своевременного их устранения и урегулирования возникших споров заказчика и поставщика. Осуществляется экспертиза на соотношение техники и её цены, оценивая стоимость исходя из дефектов, предлагается решение по устранению дефектов.

1.4 Требования к системе

1.4.1 Требования к системе в целом

1.4.1.1 Требования к структуре и функционированию системы

Определяется общей постановкой задачи задания на курсовую работу.

1.4.1.2 Требования к защите информации от несанкционированного доступа

1) Гость имеет возможность:

Посетить главную страницу и ознакомиться с основной информацией.

2) Аутентифицированный пользователь :

Имеет доступ к просмотру накладных.

3) Администратор:

Имеет полный доступ к базе данных.

1.4.2 Требования к функциям, выполняемым системой

Система должна выполнять следующие функции:

1) Система автоматически закрывает просроченные накладные и предусматривает штраф определенному сотруднику.

2) Система при вводе идентификатора выводит определенного сотрудника, его фамилию и зарплату, которая будет выплачена в зависимости от количества обработанных накладных.

3) Система вычисляет прогнозируемые доходы определенного периода.

4) Система ведёт перечень обработанных обращений клиентов.

1.4.3 Требования к видам обеспечения

1.4.3.1 Требования к техническому обеспечению

1. Процессор – AMD Ryzen™ 5 2400G with Radeon™ RX Vega 11 Graphics

Кол-во ядер CPU: 4

Кол-во потоков: 8

Базовая частота: 3.6GHz

2. Материнская плата: Gigabyte Technology Co. Ltd. B450M DS3H-CF (AM4)

3. Оперативная память: 16.0 Гб

4. Жёсткий диск: TOSHIBA HDWD110 1 ТБ

5. Видеокарта: Gigabyte NVidia GeForce GTX 1660 Super

6. Звуковая карта: Realtek High Definition Audio

7. Монитор: Samsung S24D330 24" 1920x1080

8. Оптический привод: LG HL-DT-ST DVDRAM GH24NSD1

1.4.3.2 Требования к программному обеспечению

При разработке системы использовалась Windows 10.

Microsoft Visual Studio 2019.

SQL SERVER Menagment studio 2012

ERCostructor 2.0 для создания диаграммы «сущность-связь».

1.5 Состав и содержание работ по созданию системы

Определяется этапами выполнения работы задания на курсовую работу.

1.6 Порядок контроля и приёмки системы

Определяется порядком защиты и критериями оценки работы задания на курсовую работу.

1.7. Требования к документированию

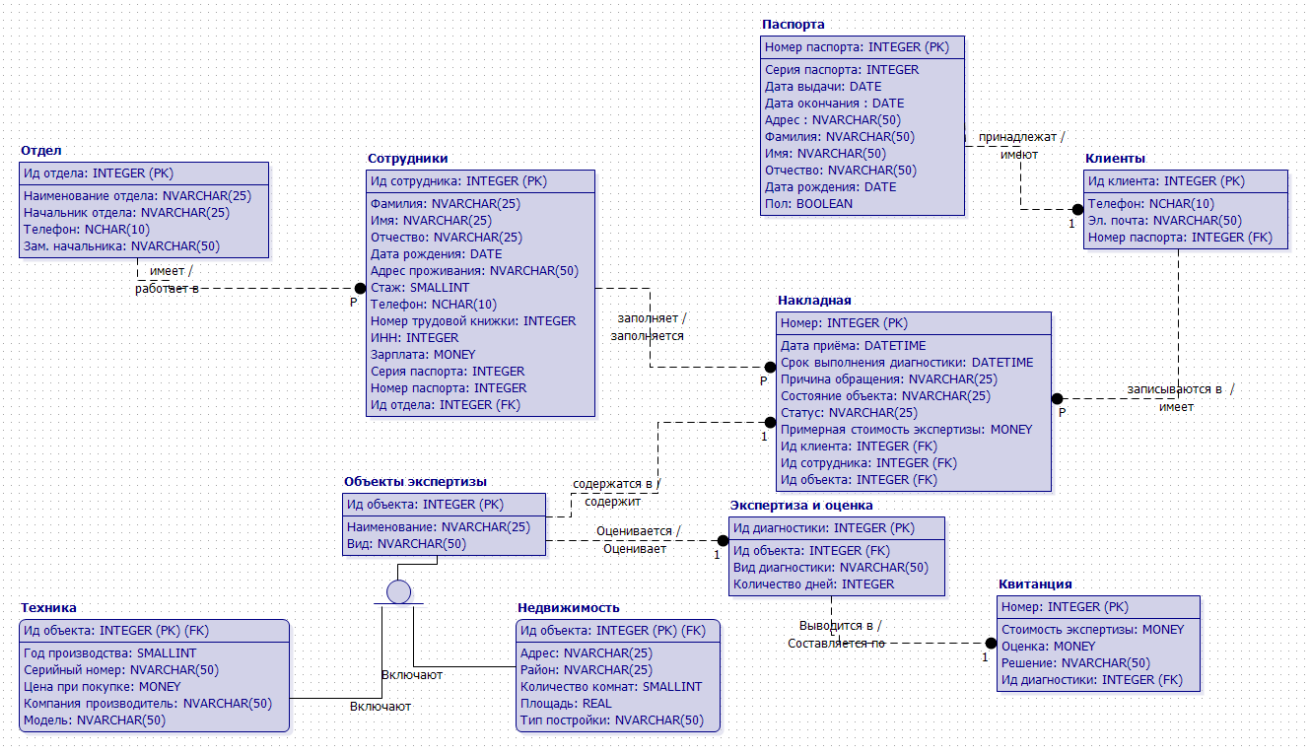
Структурные компоненты (заголовки), которые не будут включены в пояснительную записку:

- Список используемых сокращений
- Анализ концептуальной схемы

2 Информационное обеспечение системы

2.1 Концептуальная схема базы данных

2.1.1 Модель «сущность-связь»



2.1.2 Сущности и их атрибуты

Сущность **«Накладная»** содержит информацию об объекте, принимаемый на экспертизу и оценку. Атрибут «Статус» описывает состояние накладной (она может быть «закрыта», что означает: была проведена экспертиза с последующей оценкой, либо может быть «открыта», что означает: экспертизы и оценки ещё не подвергалась). Атрибут «Состояние объекта» описывает внешний вид объекта.

Сущность **«Сотрудники»** содержит информацию о всех сотрудниках работающих в «Бюро экспертизы и оценки».

Сущность **«Отдел»** содержит информацию об отделе к которому относится сотрудник (это может быть отдел недвижимости, а может быть технический отдел).

Сущность **«Клиенты»** содержит информацию о всех клиентах.

Сущность **«Паспорта»** содержит информацию об удостоверениях личности каждого клиента.

Сущность **«Объекты экспертизы»** содержит информацию о всех объектах, которым надлежит пройти стадию оценки и экспертизы. Атрибут «Вид» является уточнением данного объекта (если рассматривать технику, то это может быть стиральная машина, блендер и т.д., а если рассматривать недвижимость, то это может быть дом, квартира и т.п.).

Сущность **«Техника»** содержит уточненную информацию о технике.

Сущность **«Недвижимость»** содержит уточненную информацию о недвижимости. Атрибут «Тип постройки» описывает материал недвижимости (квартира может быть панельная, а может быть каменной; дом может быть каменным, а может быть деревянным и т.д.).

Сущность **«Экспертиза и оценка»** содержит информацию о виде выявления нарушений. Атрибут «Вид диагностики» описывает то, как будет проводиться диагностика (разбор, осмотр). Атрибут «Количество дней» описывает количество дней затраченных на экспертизу и оценку.

Сущность **«Квитанция»** содержит информацию о результате проведенной экспертизы. Атрибут «Оценка» описывает результат в денежном эквиваленте. Атрибут «Решение» описывает возможное устранение проблем, но не устраняет его (рекомендация владельцу, что нужно сделать для устранения неисправности).

2.1.3 Связи между сущностями

Отношение **«Принадлежат/Имеют»** типа один-к-одному связывает между собой сущности «Паспорта» и «Клиенты». Паспорт принадлежит одному клиенту, а каждый клиент, соответственно, имеет паспорт. Сущности «Паспорта» и «Клиенты» имеют минимальное и максимальное кардинальное число — 1, так

как паспорт всегда принадлежит какому-либо клиенту, причём только одному, а у клиента всегда есть паспорт, причём только один.

Отношение **«Содержатся в/Содержит»** типа один-к-одному связывает между собой сущности «Объекты экспертизы» и «Накладная». Объект экспертизы может входить в накладную, а накладная содержит один объект экспертизы, один объект может содержатся в одной накладной. Сущности «Объекты экспертизы» и «Накладная» имеют минимальное и максимальное кардинальное число — 1, так как один объект экспертизы может содержаться только в одной накладной, как и в одной накладной рассматривается только один объект экспертизы.

Отношение **«Оценивается/Оценивает»** типа один-к-одному связывает между собой сущности «Объекты экспертизы» и «Экспертиза и оценка». Для анализа состояния объекта используется комплекс экспертизы и оценки. Сущности «Объекты экспертизы» и «Экспертиза и оценка» имеют минимальные и максимальные кардинальные числа — 1, так как только один объект может быть исследован и оценён, и, наоборот, комплекс экспертизы и оценки может принадлежать только одному объекту.

Отношение **«Включают»** связывает сущность «Объекты экспертизы» и неполный категориальный кластер, состоящий из сущностей «Техника» и «Недвижимость». В «Бюро экспертизы и оценки» есть несколько видов объектов экспертиз, в данной модели выделяются только техника и недвижимость, поэтому кластер является неполным, так как предполагается существование других сущностей - потомков. Минимальные категориальные числа потомков равны 1.

Отношение **«Имеет/Работает в»** типа один-ко-многим связывает между собой сущности «Отдел» и «Сотрудники». В каждом отделе работают один или несколько сотрудников. Сущность «Отдел» имеет минимальное кардинальное число — 1, так как в отделе должен работать хотя бы один сотрудник, и максимальное кардинальное число – N, так как в отделе может работать неограниченное количество сотрудников. Сущность «Сотрудники» имеет минимальное и максимальное кардинальные числа — 1, так как 1 сотрудник работает только в одном отделе.

Отношение **«Выводится в/Составляется по»** типа один-к-одному связывает между собой сущности «Экспертиза и оценка» и «Квитанция». Один комплекс экспертизы и оценки выводится в квитанцию, и только одна квитанция может составляться по одной экспертизе, поэтому сущности «Экспертиза и оценка» и «Квитанция» имеют минимальное и максимальные кардинальные числа — 1.

Отношение **«Записываются в/Имеет»** типа один-ко-многим связывает между собой сущности «Клиенты» и «Накладная». Один клиент может записываться в множество накладных, одна накладная имеет одного клиента. Сущность «Клиенты» имеет минимальное кардинальное число — 1, так как для одного клиента должна быть составлена минимум одна накладная, и максимальное кардинальное число – N, так как один и тот же клиент, может фигурировать в нескольких накладных. А сущность «Накладная» имеет минимальное и максимальное кардинальные числа — 1, так как одна накладная содержит в себе одного клиента.

Отношение **«Заполняет/Заполняется»** типа один-ко-многим связывает между собой сущности «Сотрудники» и «Накладная». Один сотрудник может заполнять 1 или несколько накладных, одна накладная заполняется только одним сотрудником. Сущность «Сотрудники» имеет минимальное кардинальное число — 1, а максимальное — N, так как один сотрудник может фигурировать как в одной накладной, так и в нескольких. Сущность «Накладная» имеет максимальные и минимальные кардинальные числа равны 1, так как в одну накладную может быть записан только один сотрудник.

2.2 Внутренняя схема базы данных

2.2.1 База данных системы

1. Таблица «Users»:

```
USE [Бюро экспертизы и оценки]
    GO

    /****** Object:  Table [dbo].[Users]    Script Date: 03.06.2021 11:28:18 ******/
    SET ANSI_NULLS ON
    GO
```

```
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Users](
 [Id] [int] IDENTITY(1,1) NOT NULL,
 [Email] [nvarchar](max) NULL,
 [Password] [nvarchar](max) NULL,
 [Age] [int] NOT NULL,
 [Role] [nvarchar](max) NULL,
 CONSTRAINT [PK_dbo.Users] PRIMARY KEY CLUSTERED
(
 [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
```

2. Таблица «Паспорта»:

```
CREATE TABLE [dbo].[Паспорта](
 [Номер паспорта] [int] NOT NULL,
 [Серия паспорта] [int] NOT NULL,
 [Дата выдачи] [date] NULL,
 [Дата окончания] [date] NULL,
 [Адрес] [nvarchar](50) NULL,
 [Фамилия] [nvarchar](50) NOT NULL,
 [Имя] [nvarchar](50) NOT NULL,
 [Отчество] [nvarchar](50) NULL,
 [Дата рождения] [date] NOT NULL,
 [Пол] [bit] NOT NULL,
 CONSTRAINT [PK_Паспорта] PRIMARY KEY CLUSTERED
(
 [Номер паспорта] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Паспорта] ADD CONSTRAINT [DF_Паспорта_Адрес] DEFAULT (N'Не указан')
FOR [Адрес]
GO
ALTER TABLE [dbo].[Паспорта] ADD CONSTRAINT [DF_Паспорта_Пол] DEFAULT ((1)) FOR [Пол]
GO
```

3. Таблица «Клиенты»:

```
CREATE TABLE [dbo].[Клиенты](
 [Ид клиента] [int] NOT NULL,
 [Телефон] [nvarchar](50) NOT NULL,
 [Эл почта] [nvarchar](50) NULL,
 [Номер паспорта] [int] NOT NULL,
 CONSTRAINT [PK_Клиенты] PRIMARY KEY CLUSTERED
(
 [Ид клиента] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Клиенты] ADD CONSTRAINT [DF_Клиенты_Эл почта] DEFAULT (N'Не указа-
на') FOR [Эл почта]
GO
ALTER TABLE [dbo].[Клиенты] WITH CHECK ADD CONSTRAINT [FK_Клиенты_Паспорта] FOREIGN
KEY([Номер паспорта])
REFERENCES [dbo].[Паспорта] ([Номер паспорта])
GO
ALTER TABLE [dbo].[Клиенты] CHECK CONSTRAINT [FK_Клиенты_Паспорта]
GO
```

4. Таблица «Отдел»:

```
CREATE TABLE [dbo].[Отдел](
 [Ид отдела] [int] IDENTITY(1,1) NOT NULL,
 [Наименование отдела] [nvarchar](50) NOT NULL,
 [Начальник отдела] [nvarchar](50) NOT NULL,
 [Телефон] [nchar](10) NOT NULL,
 [Зам начальника] [nvarchar](50) NULL,
 CONSTRAINT [PK_Отдел] PRIMARY KEY CLUSTERED
(
 [Ид отдела] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

5. Таблица «Сотрудники»:

```
CREATE TABLE [dbo].[Сотрудники](
 [Ид сотрудника] [int] IDENTITY(1,1) NOT NULL,
```

```
[Фамилия] [nvarchar](50) NOT NULL,
[Имя] [nvarchar](50) NOT NULL,
[Отчество] [nvarchar](50) NULL,
[Дата рождения] [date] NOT NULL,
[Адрес проживания] [nvarchar](50) NOT NULL,
[Стаж] [smallint] NOT NULL,
[Телефон] [nchar](10) NOT NULL,
[Номер трудовой книжки] [int] NOT NULL,
[ИНН] [bigint] NULL,
[Зарплата] [money] NOT NULL,
[Серия паспорта] [int] NOT NULL,
[Номер паспорта] [int] NOT NULL,
[Ид отдела] [int] NOT NULL,
CONSTRAINT [PK_Сотрудники] PRIMARY KEY CLUSTERED
(
[Ид сотрудника] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Сотрудники] WITH CHECK ADD CONSTRAINT [Сотрудники_Отдел_FK_Rule]
FOREIGN KEY([Ид отдела])
REFERENCES [dbo].[Отдел] ([Ид отдела])
GO
ALTER TABLE [dbo].[Сотрудники] CHECK CONSTRAINT [Сотрудники_Отдел_FK_Rule]
GO
```

6. Таблица «Объекты экспертизы»:

```
CREATE TABLE [dbo].[Объекты экспертизы](
[Ид объекта] [int] NOT NULL,
[Наименование] [nvarchar](50) NOT NULL,
[Вид] [nvarchar](50) NULL,
CONSTRAINT [PK_Объекты экспертизы] PRIMARY KEY CLUSTERED
(
[Ид объекта] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

7. Таблица «Техника»:

```
CREATE TABLE [dbo].[Техника](
```

```
	[Год производства] [smallint] NULL,
	[Серийный номер] [nvarchar](50) NOT NULL,
	[Цена при покупке] [money] NULL,
	[Ид объекта] [int] NOT NULL,
	[Компания производитель] [nvarchar](50) NOT NULL,
	[Модель] [nvarchar](50) NOT NULL,
	CONSTRAINT [PK_Техника] PRIMARY KEY CLUSTERED
	(
	[Ид объекта] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY]
	GO
	ALTER TABLE [dbo].[Техника] ADD CONSTRAINT [DF_Техника_Модель] DEFAULT (N'Не указана')
FOR [Модель]
```

8. Таблица «Недвижимость»:

```
	CREATE TABLE [dbo].[Недвижимость](
	[Адрес] [nvarchar](50) NOT NULL,
	[Район] [nvarchar](50) NULL,
	[Ид объекта] [int] NOT NULL,
	[Количество комнат] [smallint] NOT NULL,
	[Площадь] [real] NOT NULL,
	[Тип постройки] [nvarchar](50) NOT NULL,
	CONSTRAINT [PK_Недвижимость] PRIMARY KEY CLUSTERED
	(
	[Ид объекта] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY]

	GO
	ALTER TABLE [dbo].[Недвижимость] ADD CONSTRAINT [DF_Недвижимость_Район] DEFAULT (N'Не
указан') FOR [Район]
	GO
	ALTER TABLE [dbo].[Недвижимость] WITH CHECK ADD CONSTRAINT [Недвижимость_Объекты экс-
пертизы_FK_Rule] FOREIGN KEY([Ид объекта])
	REFERENCES [dbo].[Объекты экспертизы] ([Ид объекта])
	ON UPDATE CASCADE
	ON DELETE CASCADE
	GO
```

```
ALTER TABLE [dbo].[Недвижимость] CHECK CONSTRAINT [Недвижимость_Объекты экспертизы_FK_Rule]
```

9. Таблица «Экспертиза и оценка»:

```
CREATE TABLE [dbo].[Экспертиза и оценка](
 [Ид диагностики] [int] NOT NULL,
 [Ид объекта] [int] NOT NULL,
 [Вид диагностики] [nvarchar](50) NOT NULL,
 [Количество дней] [int] NOT NULL,
 CONSTRAINT [PK_Экспертиза и оценка] PRIMARY KEY CLUSTERED
(
 [Ид диагностики] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
 CONSTRAINT [IX_Экспертиза и оценка] UNIQUE NONCLUSTERED
(
 [Ид объекта] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Экспертиза и оценка] WITH CHECK ADD CONSTRAINT [Экспертиза и оценка_Объекты экспертизы_FK_Rule] FOREIGN KEY([Ид объекта])
REFERENCES [dbo].[Объекты экспертизы] ([Ид объекта])
GO
ALTER TABLE [dbo].[Экспертиза и оценка] CHECK CONSTRAINT [Экспертиза и оценка_Объекты экспертизы_FK_Rule]
GO
```

10. Таблица «Квитанция»:

```
CREATE TABLE [dbo].[Квитанция](
 [Номер] [int] NOT NULL,
 [Стоимость экспертизы] [money] NOT NULL,
 [Оценка] [money] NOT NULL,
 [Решение] [nvarchar](50) NULL,
 [Ид диагностики] [int] NOT NULL,
 CONSTRAINT [PK_Квитанция] PRIMARY KEY CLUSTERED
(
 [Номер] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
```

```
 CONSTRAINT [IX_Квитанция] UNIQUE NONCLUSTERED
(
 [Ид диагностики] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Квитанция] ADD CONSTRAINT [DF_Квитанция_Решение] DEFAULT (N'Не ука-
зано') FOR [Решение]
GO
ALTER TABLE [dbo].[Квитанция] WITH CHECK ADD CONSTRAINT [Квитанция_Экспертиза и оцен-
ка_FK_Rule] FOREIGN KEY([Ид диагностики])
REFERENCES [dbo].[Экспертиза и оценка] ([Ид диагностики])
GO
ALTER TABLE [dbo].[Квитанция] CHECK CONSTRAINT [Квитанция_Экспертиза и оценка_FK_Rule]
GO
```

11. Таблица «Накладная»:

```
CREATE TABLE [dbo].[Накладная](
 [Номер] [int] NOT NULL,
 [Дата приёма] [datetime] NULL,
 [Срок выполнения диагностики] [datetime] NULL,
 [Причина обращения] [nvarchar](50) NOT NULL,
 [Состояние объекта] [nvarchar](50) NOT NULL,
 [Статус] [nvarchar](50) NOT NULL,
 [Примерная стоимость экспертизы] [money] NOT NULL,
 [Ид клиента] [int] NOT NULL,
 [Ид сотрудника] [int] NOT NULL,
 [Ид объекта] [int] NOT NULL,
 CONSTRAINT [PK_Накладная] PRIMARY KEY CLUSTERED
(
 [Номер] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
 CONSTRAINT [IX_Накладная] UNIQUE NONCLUSTERED
(
 [Ид объекта] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Накладная]  WITH CHECK ADD  CONSTRAINT [FK_Накладная_Клиенты] FOREIGN KEY([Ид клиента])
REFERENCES [dbo].[Клиенты] ([Ид клиента])
GO
ALTER TABLE [dbo].[Накладная] CHECK CONSTRAINT [FK_Накладная_Клиенты]
GO
ALTER TABLE [dbo].[Накладная]  WITH CHECK ADD  CONSTRAINT [FK_Накладная_Объекты экспертизы] FOREIGN KEY([Ид объекта])
REFERENCES [dbo].[Объекты экспертизы] ([Ид объекта])
GO
ALTER TABLE [dbo].[Накладная] CHECK CONSTRAINT [FK_Накладная_Объекты экспертизы]
GO
ALTER TABLE [dbo].[Накладная]  WITH CHECK ADD  CONSTRAINT [FK_Накладная_Сотрудники] FOREIGN KEY([Ид сотрудника])
REFERENCES [dbo].[Сотрудники] ([Ид сотрудника])
GO
ALTER TABLE [dbo].[Накладная] CHECK CONSTRAINT [FK_Накладная_Сотрудники]
GO
```

Хранимые процедуры и функции:

1. Процедура procEditNaklad заполняет в таблице «Накладная» атрибут «Статус» и в таблице «Сотрудники» атрибут «Зарплата». Процедура находит не закрытые «Накладные» данного сотрудника, закрывает их, а также, уменьшает зарплату тому сотруднику, если таких «Накладных» более 1. Сотрудник выбирается исходя из серии и номера паспорта (@Series, @Nomer):

```
CREATE PROCEDURE [dbo].[procEditNaklad]
(
 @Series INT,
 @Nomer  INT
)
AS
BEGIN

  -- Находим Id сотрудника по серии и номеру паспорта
  DECLARE @idSotr INT = (SELECT [Ид сотрудника] FROM dbo.Сотрудники WHERE[Серия паспорта] = @Series AND [Номер паспорта] = @Nomer);

  -- по Id находим фамилию сотрудника
  DECLARE @Familia NVARCHAR(50) = (SELECT Фамилия FROM Сотрудники WHERE [Ид сотрудника] = @idSotr);
```

```
	-- по Id и дате определяем количество не закрытых накладных данного сотрудника
	DECLARE @kolvoNeZakr INT = (SELECT COUNT(*) FROM dbo.Накладная WHERE [Срок выполнения диагностики] <= CONVERT (date,GETDATE()) AND LOWER(Статус) = 'открыт' AND [Ид сотрудника] = @idSotr);

	-- находим по Id стаж сотрудника для вычисления коэффициента штрафа
	DECLARE @stag INT = (SELECT Стаж FROM Сотрудники WHERE [Ид сотрудника] = @idSotr);

	-- если не закрытые накладные найдены, тогда закрываем
	IF (@kolvoNeZakr) > 0
	BEGIN
		UPDATE dbo.Накладная SET Статус = 'Закрыт' WHERE [Срок выполнения диагностики] <= CONVERT(date,GETDATE()) AND LOWER(Статус) = 'открыт' AND [Ид сотрудника] = @idSotr;
	END;

	DECLARE @koefShtrafa MONEY;
	DECLARE @ZarplataEdit MONEY;

	-- уменьшаем зарплату в зависимости от количества не закрытых накладных
	IF (@kolvoNeZakr > 1)
	BEGIN
		-- коэфициент штрафа
		SET @koefShtrafa = ( SELECT (@stag/@kolvoNeZakr) FROM Сотрудники WHERE [Ид сотрудника] = @idSotr);

		-- зарплата измененная коэфициентом
		SET @ZarplataEdit  = ( SELECT Зарплата*0.99*@koefShtrafa FROM Сотрудники WHERE [Ид сотрудника] = @idSotr);
		PRINT 'Зарплата на данный месяц ' + CONVERT(varchar(20),@ZarplataEdit);

		UPDATE dbo.Сотрудники SET Зарплата = @ZarplataEdit WHERE [Ид сотрудника] = @idSotr;
	END;
	ELSE
	BEGIN
		PRINT 'У сотрудника ' + @Familia + ' все накладные закрыты';
	END;
END
```

2. Функция funcZarplata выводит таблицу с идентификатором конкретного сотрудника, фамилией сотрудника и зарплатой, которая будет выплачена в зависимости от количества обработанных экспертиз. Сотрудник выбирается по @Id:

```
CREATE FUNCTION [dbo].[funcZarplata]
      (
       @Id INT
      )
      RETURNS @result TABLE (Ид_сотрудника INT, [Фамилия] NVARCHAR(50), [Зарплата] MONEY)
      AS
      BEGIN

       DECLARE @countNakl INT;
       SELECT @countNakl = COUNT(*) FROM Накладная WHERE Накладная.[Ид сотрудника] = @Id;

      /*
       Сводка данных по выбранному id сотрудника
      */
       IF (@countNakl > 4)
       BEGIN
              INSERT INTO @result
              SELECT Накладная.[Ид сотрудника], Сотрудники.Фамилия, Сотрудники.Зарплата * 1.25
FROM Накладная, Сотрудники
              WHERE Накладная.[Ид сотрудника] = @Id AND Сотрудники.[Ид сотрудника] = @Id
              GROUP BY Накладная.[Ид сотрудника], Сотрудники.Фамилия, Сотрудники.Зарплата;
       END;
       ELSE
       BEGIN
              INSERT INTO @result
              SELECT Накладная.[Ид сотрудника], Сотрудники.Фамилия, Сотрудники.Зарплата FROM
Накладная, Сотрудники
              WHERE Накладная.[Ид сотрудника] = @Id AND Сотрудники.[Ид сотрудника] = @Id
              GROUP BY Накладная.[Ид сотрудника], Сотрудники.Фамилия, Сотрудники.Зарплата;
       END;

       RETURN

      END;
```

3. Функция funcProgDohod выводит таблицу с количество обработанных накладных и прогнозируемого дохода за период. Прогнозируемый доход подсчитывается с помощью двух дат. Первая дата с которого происходит подсчёт - @date1, вторая дата, до какого срока происходит подсчёт - @date2:

```
CREATE FUNCTION [dbo].[funcProgDohod]
(
 @date1 DATETIME,
 @date2 DATETIME
)
RETURNS @result TABLE ([Количество обработанных накладных] INT, [Прогнозируемый доход] MONEY)
AS
BEGIN

 DECLARE @sumExp MONEY;
 DECLARE @countNakl INT;


 select @sumExp = SUM([Примерная стоимость экспертизы])  from Накладная where Накладная.[Дата приёма] between @date1 and @date2;
 select @countNakl = count(*) from Накладная where Накладная.[Дата приёма] between @date1 and @date2;

 INSERT TOP (1) INTO @result
              SELECT @countNakl, @sumExp FROM Накладная
              WHERE  Накладная.[Дата приёма] between @date1 and @date2
              GROUP BY Накладная.Номер, [Примерная стоимость экспертизы];

 return
END
```

3. Алгоритмическое обеспечение системы

3.1 Общая характеристика: назначение алгоритма

Данный алгоритм обновляет записи в таблицах «Накладная» и «Сотрудники», при помощи хранимой процедуры исходя из заданных параметров.

3.2 Используемые данные

Таблица «Накладная»:

1 Атрибут «Срок выполнения диагностики»

2 Атрибут «Статус»

3 Атрибут «Ид сотрудника»

Таблица «Сотрудники»

4 Атрибут «Ид сотрудника»

5 Атрибут «Фамилия»

6 Атрибут «Стаж»

7 Атрибут «Зарплата»

8 Атрибут «Серия паспорта»

9 Атрибут «Номер паспорта»

3.3 Результаты выполнения: перечень таблиц базы данных и их атрибутов, экранных форм, формируемых или изменяемых в результате выполнения алгоритма.

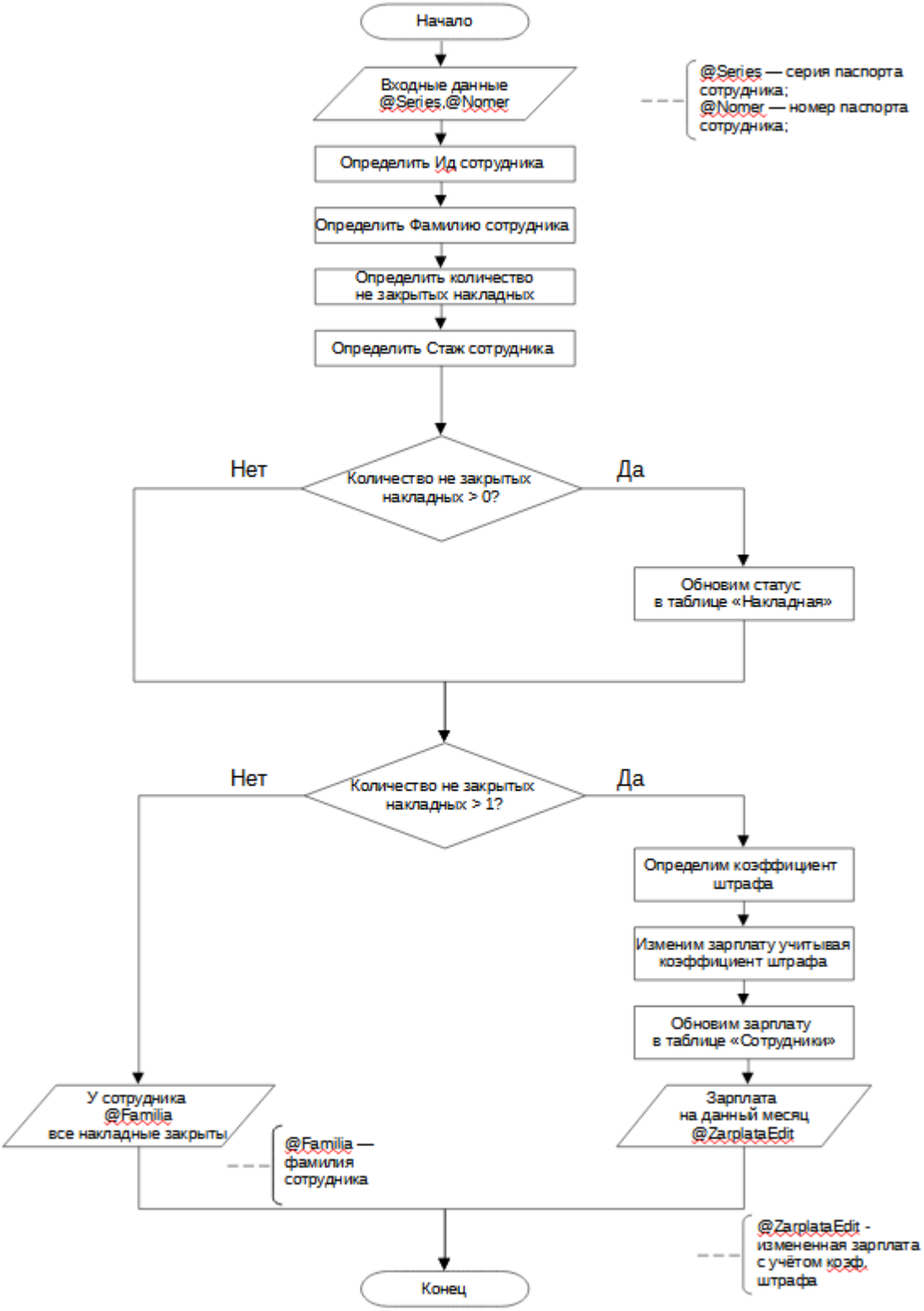
Таблица «Накладная»:

- Атрибут «Статус»

Таблица «Сотрудники»

- Атрибут «Зарплата»

3.4 Логическое описание:



4 Прикладное программное обеспечение системы

4.1 Общая характеристика прикладного программного обеспечение

1. Инструментальная среда разработки – Visual Studio Community 2019
2. СУБД – Microsoft SQL Server Management Studio 12
3. Моделирование данных выполнено с помощью средства автоматизации ERConstructor 2.0.
4. Технология разработки – ASP.NET MVC.
5. Технология доступа к данным – ADO.NET Entity Framework.

Обработка исключительных ситуаций реализована при помощи атрибутов валидации и нескольких представлений с пометкой Error.

В таблицах можно просматривать, редактировать, добавлять и удалять записи.

Присутствует аутентификация, которая осуществляется по логину и паролю, проверка происходит через базу данных.

Поиск по базе данных:

- Поиск клиента по имени и фамилии в «Паспорт».
- Поиск сотрудника по имени и фамилии в «Сотрудники».

Фильтрация:

1) Вывод квитанции по определенному промежутку стоимости экспертизы в «Квитанции».

2) Вывод только открытых накладных с помощью CheckBox в «Накладная»

3) Вывод сотрудников по отделам с помощью DropDownList в «Сотрудники».

Аутентификация:

Пользователю предлагается войти в систему, при успешной аутентификации он получает роль «user». Далее в зависимости от введенного логина и пароля

пользователь получает определенные привилегии доступа в зависимости от его роли. Если он заходит с ролью Администратор, то ему представляется список возможностей для администратора,.

База данных содержит:

1) Количество таблиц – 11.

2) Количество атрибутов – 50.

3) Количество типов связей — 3.

4) Количество хранимых процедур – 1.

Процедура procEditNaklad работает с полями в таблицах «Накладная» и «Сотрудники»: «Статус» и «Зарплата». Обновляет «Статус» в таблице «Накладная» и «Зарплата» в таблице «Сотрудники».

5) Количество функций — 2(табличные функции).

1. Функция funcZarplata выводит таблицу с идентификатором конкретного сотрудника, фамилией сотрудника и зарплатой, которая будет выплачена в зависимости от количества обработанных экспертиз. Сотрудник выбирается по @Id

2. Функция funcProgDohod выводит таблицу с количество обработанных накладных и прогнозируемого дохода за период. Прогнозируемый доход подсчитывается с помощью двух дат. Первая дата с которого происходит подсчёт - @date1, вторая дата, до какого срока происходит подсчёт - @date2:

6) Количество индексов - 10.

7) Значений по умолчанию - 6

4.2 Структура и состав прикладного программного обеспечения

Обозреватель решений в проекте:

Properties – папка, которая отвечает за настройку сборки. Содержит в себе сведения о сборке, версии сборки, номер сборки и номер редакции;

Ссылки – папка, в которой находятся ссылки на сборки, используемые в проекте.

AppData – папка с файлами баз данных.

AppStart – папка с файлами, содержащая некоторую конфигурацию для сборки проекта.

Content – папка, содержащая стили и скрипты, используемые в проекте.

Controllers – папка, которая содержит контроллеры проекта.

Fonts – содержит шрифты, используемые в проекте;

Images – содержит изображение картинки, используемой в проекте.

Models – папка с файлами моделей данных.

Scripts – папка, с файлами JavaScript.

Views – папка со всеми представлениями в проекте.

В корневой директории проекта находятся файлы .edmx – модели данных ADO.NET EDM, используемые в проекте.

Контроллеры:

HomeController отвечает за вывод главной страницы и страницы для гостей. Значимых строк кода – 2.

АутентификацияController нужен для вывода страницы аутентификации и самой аутентификации. Значимых строк кода – 20.

КвитанцияController представляет сведения о квитанциях. Есть возможность просматривать, добавлять, редактировать и удалять записи. Также есть фильтрация по диапазону по полю «Стоимость экспертизы». Значимых строк кода – 51.

КлиентыController предоставляет список клиентов. Есть возможность просматривать, добавлять, редактировать и удалять записи. Значимых строк кода – 47.

НакладнаяController предоставляет список накладных, также представления для пользователя(users). Есть возможность просматривать, добавлять, редактировать и удалять записи. Также присутствует фильтрация по 1 полю. Помимо

этого, содержит в себе реализацию процедуры. А также табличную функцию. Значимых строк кода – 77.

Начальная_страницаController отвечает за вывод страницы администратора и страницы пользователя(users). Значимых строк кода — 2.

НедвижимостьController предоставляет сведения недвижимости. Есть возможность просматривать, добавлять, редактировать и удалять записи. Значимых строк кода – 47.

Объекты_экспертизыController предоставляет данные о объектах экспертизы. Есть возможность просматривать, добавлять, редактировать и удалять записи. Значимых строк кода – 50.

ОтделController предоставляет данные об отделах, к которым «привязаны» сотрудники.. Есть возможность просматривать, добавлять, редактировать и удалять записи. Значимых строк кода – 42.

ПаспортаController предоставляет паспортные данные клиента. Есть возможность просматривать, добавлять, редактировать и удалять элементы. Содержит в себе поиск, осуществляемый по 2 полям. Значимых строк кода – 47.

СотрудникиController предоставляет записи всех сотрудников. Есть возможность просматривать, добавлять, редактировать и удалять элементы. Содержит в себе поиск по 2 полям, табличную функцию и фильтр, осуществляемый с помощью DropDownList. Значимых строк кода – 61.

ТехникаController предоставляет записи техники. Есть возможность просматривать, добавлять, редактировать и удалять элементы. Значимых строк кода – 47.

Экспертиза_и_оценкаController необходим для предоставления записей экспертизы и оценки. Есть возможность просматривать, добавлять, редактировать и удалять элементы. Значимых строк кода – 47.

4.3 Особенности реализации и сопровождения

Благодаря типовым контроллерам MVC к данным базы данных был осуществлен простой и удобный доступ. Интерфейс не является тяжёлым.

Реализация аутентификации: для реализации создавалась в SQL SERVER’ е таблица, которая хранит в себе данные о пользователе(логин, пароль, роль пользователя и др.). Атрибут «Роль пользователя» является необходимым для определения того, к какому типу относится пользователь: admin или user. Для каждой роли, я реализовал соответствующее представление. И после прохождения аутентификации пользователь переопределялся на своё представление(представление для admin, либо — для user).

5 Руководство пользователя

5.1 Общие сведения

Система представляет из себя обобщенную модель «Бюро экспертизы и оценки». Она предоставляет необходимую пользователю информацию. Также имеет пользовательские функции: обновление статус в накладной и изменения зарплаты по серии и номеру паспорта сотрудника, подсчёт зарплаты сотрудника, подсчёт прогнозируемого дохода.

5.2 Порядок и особенности работы

1. Запустив проект, пользователь увидит главную страницу(рис. 5.1). Она предоставляет из себя страничку визитку, можно перейти к более подробной информации нажав на кнопку «Узнать подробнее». Либо же сразу пройти аутентификацию и войти под своим логином. Регистрации не существует, т. к. сотрудники сами заполняют данные клиента и выдают ему логин с паролем.

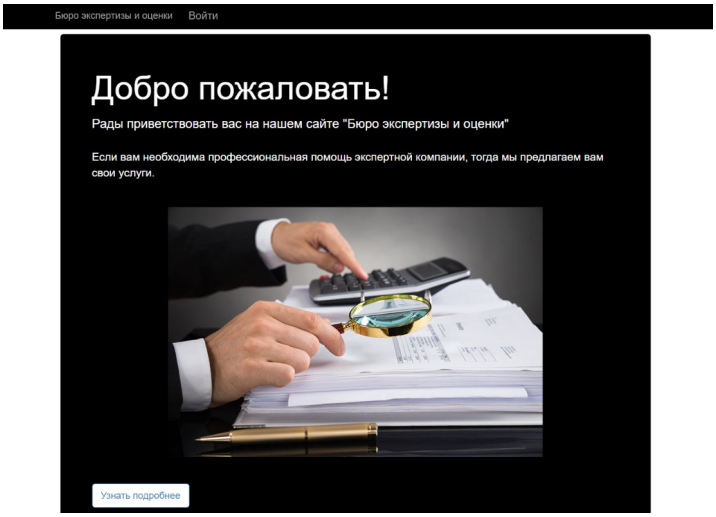


Рисунок 5.1 – Главная страница

2. Нажмём на кнопку «Узнать подробнее»(рис. 5.2). Здесь мы может наблюдать информацию о «Бюро экспертизы и оценки». В которой прописано, что такое «Бюро экспертизы и оценки», какие услуги предоставляет, в чём преимущество.

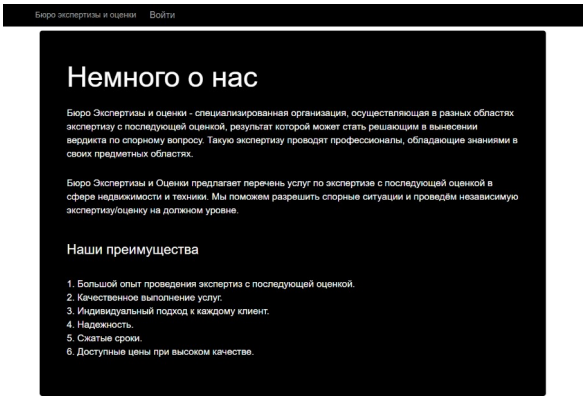


Рисунок 5.2 – Узнать подробнее.

3. Далее перейдём во вкладку «Войти». Перед пользователем возникает форма для заполнения(рис. 5.3). В зависимости от данных он вводит их. И переходит либо под ролью администратора (admin), либо под ролью пользователя (users).

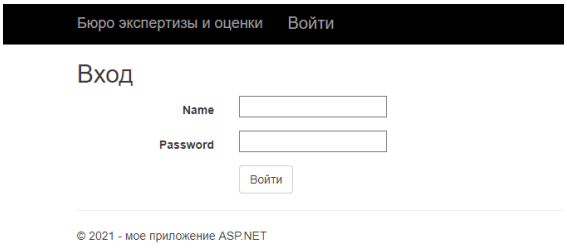


Рисунок 5.3 – Аутентификация.

4. После успешного входа, к примеру, за пользователя (users). Появляется страничка пользователя(рис. 5.4), означает, что аутентификация прошла без проблем, и вошла на личную страничку пользователя. На данной страничке есть только доступ к накладным и то данная таблица, доступна лишь для чтения.

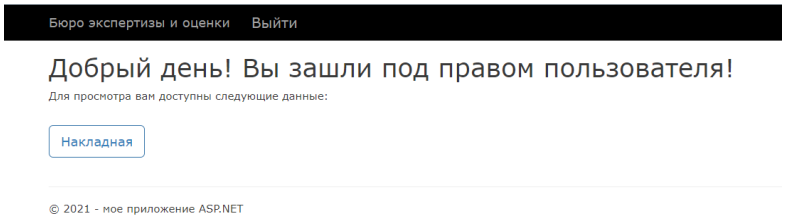


Рисунок 5.4 – Аутентификация и вход за пользователя.

5. Переходим к таблице «Накладная»(рис. 5.5). Как можно увидеть доступ: «режим — чтения»

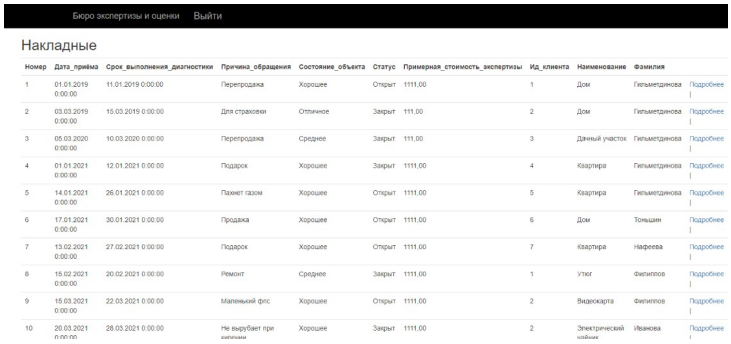


Рисунок 5.5 – Доступ «режим — чтения».

6. В таблице справа нажимаем на «Подробнее», чтобы узнать больше информации о определенной накладной(рис. 5.6).

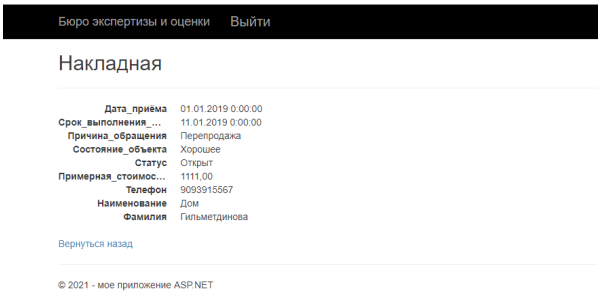


Рисунок 5.6 – Подробнее.

7. Нажимаем на вкладку выйти. И делаем тоже самое, но заходим под правом администратора. Пользователям с данной ролью разрешены все виды изменений(рис. 5.7).

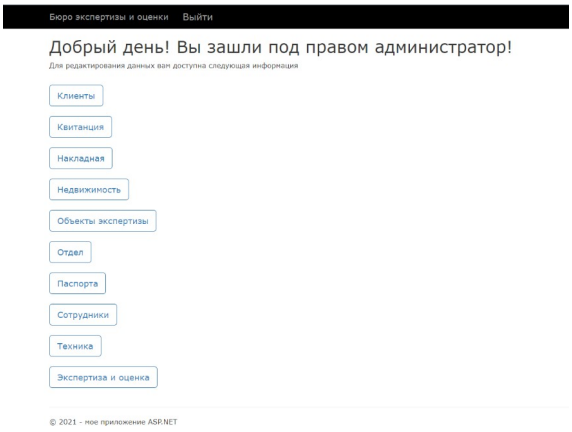


Рисунок 5.7 – Аутентификация
и вход пользователя под правом администратора.

8. Основные возможности администратора системы можно также рассмотреть на примере «Паспорта»(рис. 5.8). Такой пользователь может: добавлять новые паспорта (рис. 5.9), изменять информацию о паспортах (рис. 5.10), удалять информацию о них(рис 5.11).

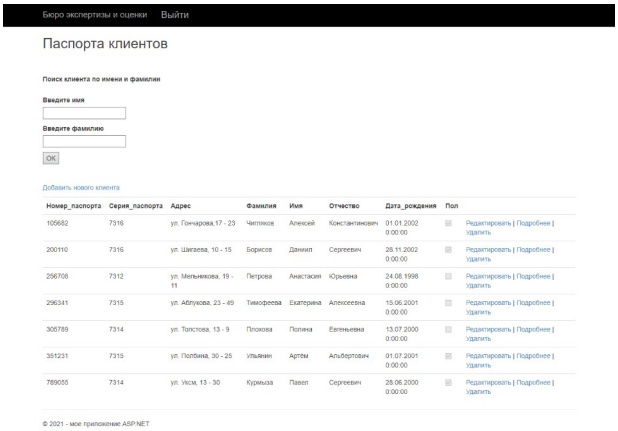


Рисунок 5.8 – Таблица «Паспорта».

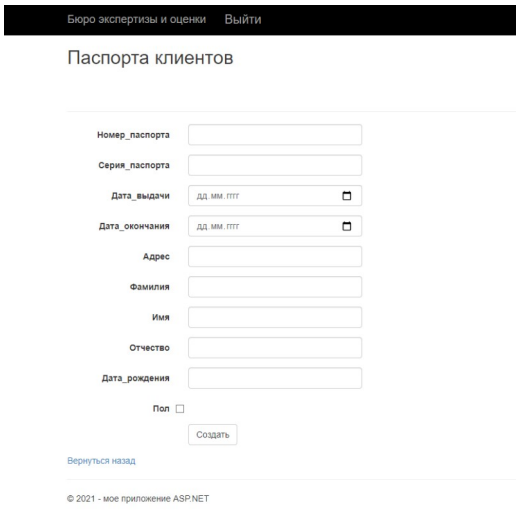


Рисунок 5.9 – Добавление.

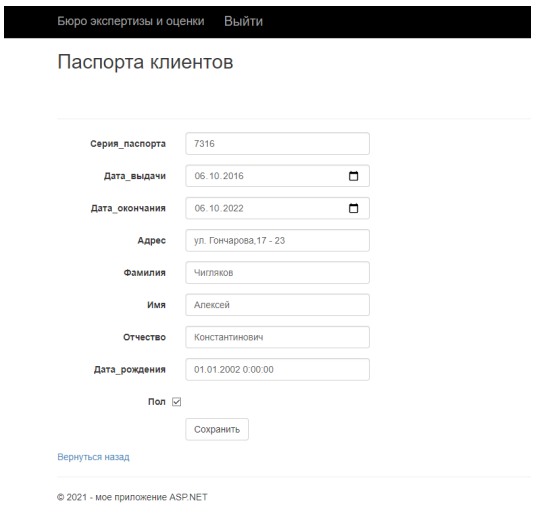


Рисунок 5.10 – Редактирование.

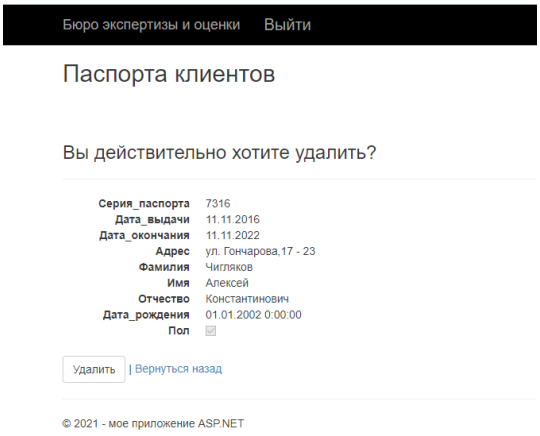


Рисунок 5.11 – Удаление.

Также присутствует поиск клиента по полям «Имя», «Фамилия» (рис. 5.12). В качестве результата будет строка данного клиента.

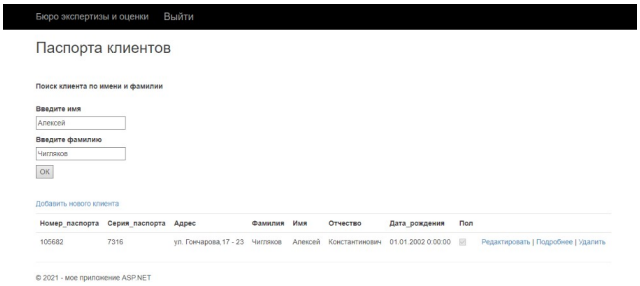


Рисунок 5.12 – Поиск.

9. Также рассмотрим сущность «Квитанция». Содержится вся основная информация экспертизы и оценки(рис. 5.13).

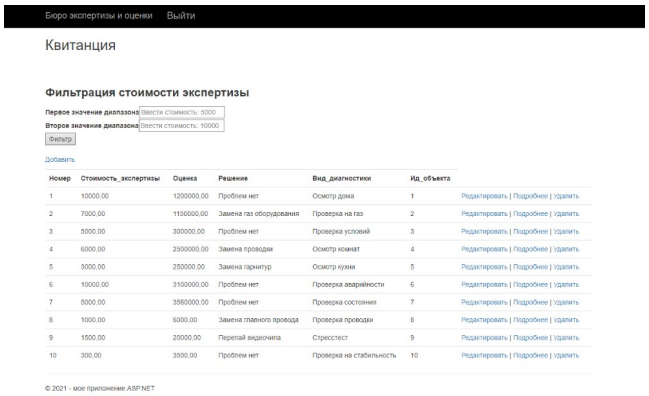


Рисунок 5.13 – Таблица «Квитанция».

А также здесь можно отфильтровать записи по стоимости экспертизы(рис. 5.14)

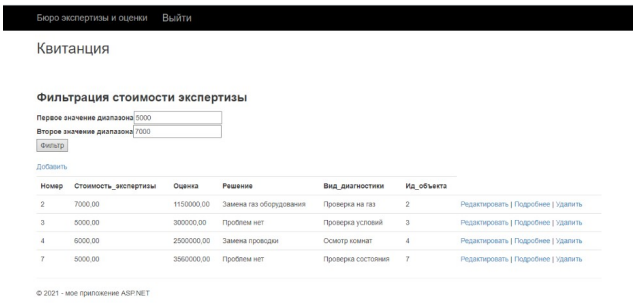


Рисунок 5.14 – Фильтрация.

10. В таблице «Сотрудники» (рис. 5.15) реализован поиск сотрудника по имени и фамилии(рис. 5.16), фильтрация сотрудников по отделам из выпадающего списка(рис. 5.17), а также функция вывода зарплаты у конкретного сотрудника в зависимости от обработанных заказов(рис. 5.18).

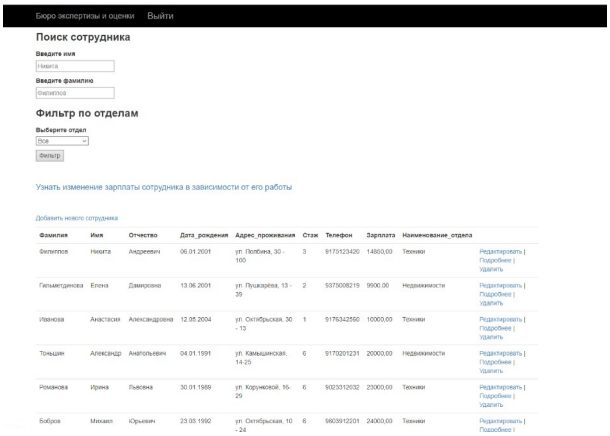


Рисунок 5.15 – Таблица «Сотрудники».

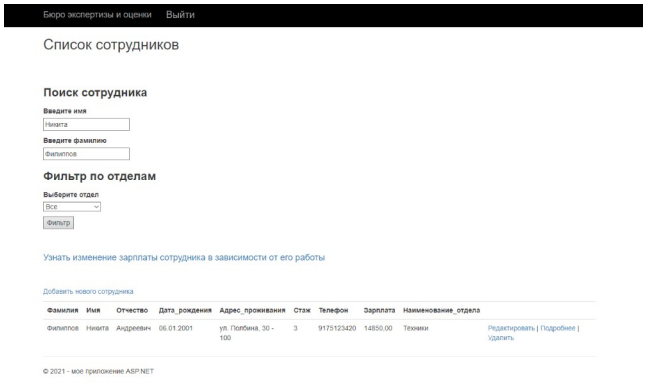


Рисунок 5.16 – Поиск.

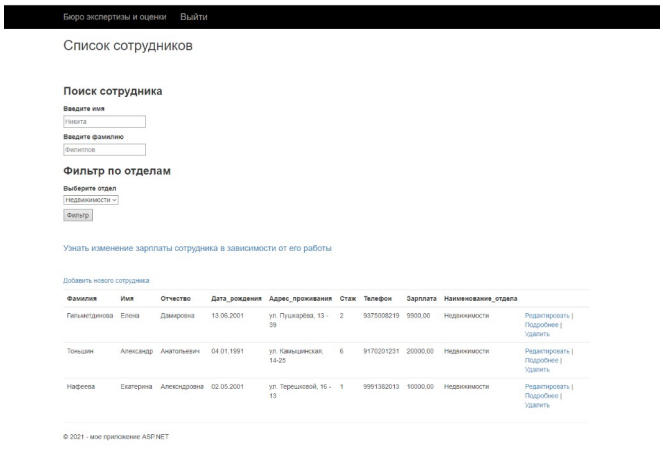


Рисунок 5.17 – Фильтрация.

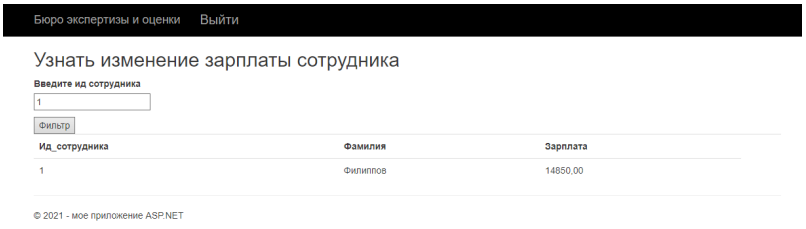


Рисунок 5.18 – Функция.

11. В таблице «Накладная»(рис. 5.19) реализована операция фильтрации для отображения открытых накладных(рис. 5.20), процедура по закрытию накладных у определенного сотрудника(рис. 5.21 - 5.22), а также функция по вычислению прогнозируемого дохода(рис. 5.23).

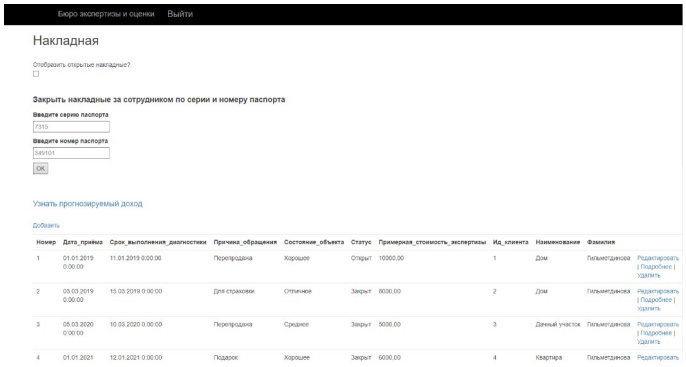


Рисунок 5.19 – Таблица «Накладная».

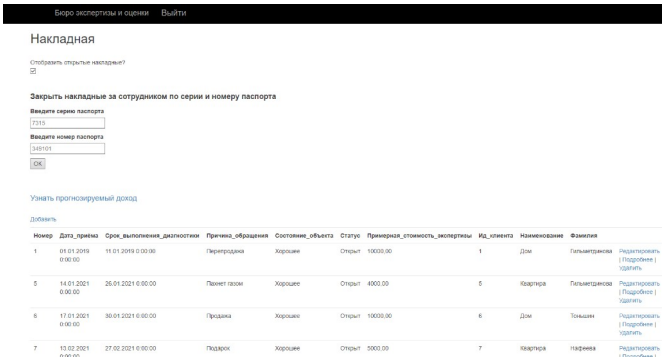


Рисунок 5.20 – Фильтрация.

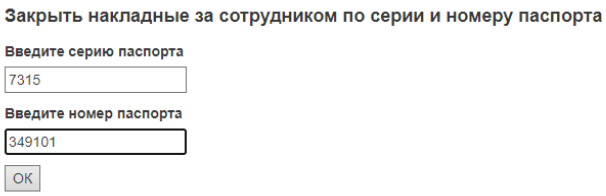


Рисунок 5.21 – Процедура по закрытию накладных.

У сотрудника с фамилией Филиппов, были открытые накладные, процедура их закрыла.

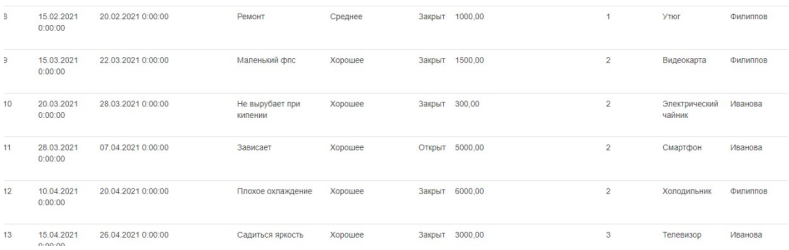


Рисунок 5.22 – Процедура по закрытию накладных..

Для вычисления прогнозируемого дохода перейдем по ссылке: «Узнать прогнозируемый доход».

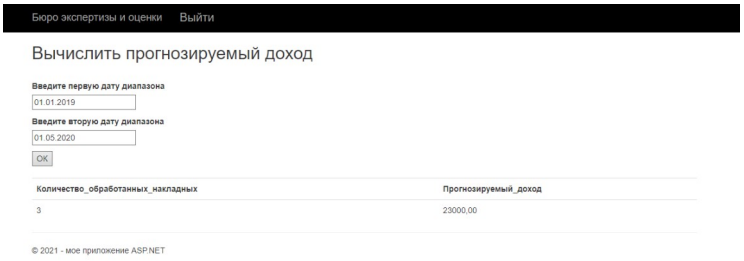


Рисунок 5.23 – Функция вычисления прогнозируемого дохода**.**

5.3 Исключительные ситуации

1. Исключительная ситуация может возникнуть при попытке аутентификации. Если пользователь введёт некорректные данные, система оповестит его об этом.
2. Также исключительная ситуация может возникнуть в том случае, когда пользователь оставит какое-либо обязательное поле в любой из таблиц незаполненным при редактировании информации или добавлении новых данных. Система не даст совершить изменения в базе данных.
3. Страница ошибки будет отображена пользователю, если он попытается добавить нового клиента или изменить информацию о текущем так, что идентификатор клиента будет уже занят кем-либо из клиентов в базе данных, либо если не будет паспорта нового клиента, то тоже выдаст страницу ошибки.
4. Страница ошибки будет отображена пользователю, если он попытается удалить сотрудника, который будет присутствовать в любой из накладных. Решение простое: нужно убрать данного сотрудника из накладной, и только затем его удалить.

Заключение

Данная работа оформлена в как можно более удобном виде для пользователя, интерфейс сделан простым и понятным для использования. Все необходимые требования в соответствии с этой работой были соблюдены.

В ходе работы я периодически прибегал к помощи интернет-форумов ввиду невнимательности или неосведомленности некоторых пунктов, однако, благодаря этим трудностям, я научилась этих ошибок избегать. Долгое время у меня заняло продумывание деталей логических процедур и функций, и особенно аутентификация, но все же я смогли осуществить свои задумки и закончить работу.

Список использованных источников

1. АНО «Судебный Эксперт» - судебная и внесудебная независимая экспертиза. Список проводимых экспертиз : сайт. - URL: https://sudexpa.ru/expertises/ (дата обращения: 04.02.2021). – Текст : электронный

2. Бюро независимой экспертизы : сайт. - URL: http://ul-expert.ru (дата обращения: 28.05.2021). – Текст : электронный

3. ГОСТ Р.7.0.100-2018 Система стандартов по информации, библиотечному и издательскому делу - предоставил мне основные данные по оформлению. (дата обращения: 04.02.2021). – Текст : электронный

4. Документация по ASP.NET MVC// Майкрософт URL: https://www.microsoft.com/ru-ru (дата обращения: 28.05.2021). – Текст : электронный

5. Областное Бюро Экспертизы и Оценки : сайт. - URL: http://expert-ws.ru (дата обращения: 02.04.2021). – Текст : электронный

6. Отосина В. Н. Организация и проведение экспертизы и оценки качества товаров / В.Н. Отосина. – Москва : КНОРУС, 2019. – 210 с. – ISBN: 978-5-406-06867-0. – URL: https://avidreaders.ru/book/organizaciya-i-provedenie-ekspertizy-i-ocenki.html (дата обращения: 30.05.2021). – Текст : электронный.

7. Ростовский центр судебных экспертиз : сайт. —URL: https://rostexpert.ru/ (дата обращения: 04.02.2021). – Текст : электронный.

8. Руководство по ASP.NET MVC 5 // URL: https://metanit.com/sharp/mvc5/ (дата обращения: 01.06.2021). – Текст : электронный

9. AG – полезная информация для вас : сайт. -URL: https://autogear.ru/article/289/380/ekspertnoe-byuro---eto-organizatsiya-kotoraya-pridet-na-pomosch/ (дата обращения: 30.05.2021). – Текст : электронный

Приложение А. Исходные тексты программных модулей

HomeController

```
using System;
      using System.Collections.Generic;
      using System.Linq;
      using System.Web;
      using System.Web.Mvc;

      namespace DB_BureauExpertiseAndEvaluation.Controllers
      {
          public class HomeController : Controller
          {
              public ActionResult Index()
              {
                  return View();
              }

              public ActionResult IndexAfter()
              {
                  return View();
              }
          }
}
```

АутентификацияController

```
using DB_BureauExpertiseAndEvaluation.Models;
      using System;
      using System.Collections.Generic;
      using System.Linq;
      using System.Web;
      using System.Web.Mvc;
      using System.Web.Security;
      using System.Web.UI;
      using System.Web.UI.WebControls;

      namespace DB_BureauExpertiseAndEvaluation.Controllers
      {
          public class АутентификацияController : Controller
```

```
        {
            // GET: Аутентификация
            public ActionResult Index()
            {
                return View();
            }

            public ActionResult Login()
            {
                return View();
            }

            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Login(UserLogin model)
            {
                if (ModelState.IsValid)
                {
                    // поиск пользователя в бд
                    User user = null;
                    string role = string.Empty;

                    int id;
                    using (UserContext db = new UserContext())
                    {
                        try
                        {
                                 user = db.Users.FirstOrDefault(u => u.Email == model.Name &&
u.Password == model.Password);
                                role = db.Users.Where(n => n.Email == model.Name).Select(s =>
s.Role).Single();
                                 id = db.Users.Where(n => n.Email == model.Name).Select(s =>
s.Id).Single();
                        }
                        catch
                        {

                        }
                    }
                    if (user != null)
                    {
                        if (role == "admin")
                        {
```

```
                            return RedirectToAction("IndexAdmin", "Начальная_страница");
                        }
                        else if(role == "user")
                        {
                            return RedirectToAction("IndexUser", "Начальная_страница");
                        }
                        else
                        {
                              ViewBag.Message = string.Format("Hello {0}.\\nCurrent Date and
Time: {1}", "artem:", DateTime.Now.ToString());
                        }

                        //FormsAuthentication.SetAuthCookie(model.Name, true);
                        //return RedirectToAction("Index", "Home");
                    }
                    else
                    {
                         ModelState.AddModelError("", "Пользователя с таким логином и паролем
нет");
                    }
                }

                return View(model);
            }

            public ActionResult Register()
            {
                return View();
            }

            public ActionResult Error()
            {
                return View();
            }

            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Register(UserRegister model)
            {
                if (ModelState.IsValid)
                {
                    User user = null;
                    using (UserContext db = new UserContext())
```

```
                    {
                        user = db.Users.FirstOrDefault(u => u.Email == model.Name);
                    }
                    if (user == null)
                    {
                        // создаем нового пользователя
                        using (UserContext db = new UserContext())
                        {
                                    db.Users.Add(new User { Email = model.Name, Password =
model.Password });
                            db.SaveChanges();

                              user = db.Users.Where(u => u.Email == model.Name && u.Password
== model.Password).FirstOrDefault();
                        }
                        // если пользователь удачно добавлен в бд
                        if (user != null)
                        {
                            FormsAuthentication.SetAuthCookie(model.Name, true);
                            return RedirectToAction("Index", "Home");
                        }
                    }
                    else
                    {
                        ModelState.AddModelError("", "Пользователь с таким логином уже суще-
ствует");
                    }
                }

                return View(model);
            }

            public ActionResult Logoff()
            {
                FormsAuthentication.SignOut();
                return RedirectToAction("Index", "Home");
            }
        }
}
```

КвитанцияController

```
using System;
      using System.Collections.Generic;
      using System.Data;
      using System.Data.Entity;
      using System.Linq;
      using System.Net;
      using System.Web;
      using System.Web.Mvc;
      using DB_BureauExpertiseAndEvaluation;

      namespace DB_BureauExpertiseAndEvaluation.Controllers
      {
          public class КвитанцияController : Controller
          {
                   private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

              // GET: Квитанция
              public ActionResult Index()
              {
                  var квитанция = db.Квитанция.Include(к => к.Экспертиза_и_оценка);
                  return View(квитанция.ToList());
              }

              [HttpPost]
              public ActionResult Index(decimal ?firstValue, decimal ?secondValue)
              {
                  var order = db.Квитанция.AsEnumerable();

                  if (firstValue != null && secondValue != null)
                  {
                       order = db.Квитанция.Where(n => n.Стоимость_экспертизы >= firstValue &&
n.Стоимость_экспертизы <= secondValue).Select(k => k);
                  }
                  return View(order.ToList());
              }

              // GET: Квитанция/Details/5
              public ActionResult Details(int? id)
              {
                  if (id == null)
```

```
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Квитанция квитанция = db.Квитанция.Find(id);
            if (квитанция == null)
            {
                return HttpNotFound();
            }
            return View(квитанция);
        }

        // GET: Квитанция/Create
        public ActionResult Create()
        {
             ViewBag.Ид_диагностики = new SelectList(db.Экспертиза_и_оценка, "Ид_диагно-
стики", "Вид_диагностики");
            return View();
        }

        public ActionResult Error()
        {
            return View();
        }

        // POST: Квитанция/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                    public    ActionResult    Create([Bind(Include    =
"Номер,Стоимость_экспертизы,Оценка,Решение,Ид_диагностики")] Квитанция квитанция)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Квитанция.Add(квитанция);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
```

```
                    return RedirectToAction("Error");
                }
            }

             ViewBag.Ид_диагностики = new SelectList(db.Экспертиза_и_оценка, "Ид_диагно-
стики", "Вид_диагностики", квитанция.Ид_диагностики);
            return View(квитанция);
        }

        // GET: Квитанция/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Квитанция квитанция = db.Квитанция.Find(id);
            if (квитанция == null)
            {
                return HttpNotFound();
            }
             ViewBag.Ид_диагностики = new SelectList(db.Экспертиза_и_оценка, "Ид_диагно-
стики", "Вид_диагностики", квитанция.Ид_диагностики);
            return View(квитанция);
        }

        // POST: Квитанция/Edit/5
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
         public ActionResult Edit([Bind(Include = "Номер,Стоимость_экспертизы,Оценка,Ре-
шение,Ид_диагностики")] Квитанция квитанция)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(квитанция).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
```

```
                    catch
                    {
                        return RedirectToAction("Error");
                    }
                }
                 ViewBag.Ид_диагностики = new SelectList(db.Экспертиза_и_оценка, "Ид_диагно-
стики", "Вид_диагностики", квитанция.Ид_диагностики);
                return View(квитанция);
            }

            // GET: Квитанция/Delete/5
            public ActionResult Delete(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Квитанция квитанция = db.Квитанция.Find(id);
                if (квитанция == null)
                {
                    return HttpNotFound();
                }
                return View(квитанция);
            }

            // POST: Квитанция/Delete/5
            [HttpPost, ActionName("Delete")]
            [ValidateAntiForgeryToken]
            public ActionResult DeleteConfirmed(int id)
            {
                Квитанция квитанция = db.Квитанция.Find(id);
                try
                {
                    db.Квитанция.Remove(квитанция);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
                }
            }
```

```
            protected override void Dispose(bool disposing)
            {
                if (disposing)
                {
                    db.Dispose();
                }
                base.Dispose(disposing);
            }
        }
    }
```

КлиентыController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class КлиентыController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Клиенты
            public ActionResult Index()
            {
                var клиенты = db.Клиенты.Include(к => к.Паспорта);
                return View(клиенты.ToList());
            }


            // GET: Клиенты/Details/5
            public ActionResult Details(int? id)
            {
                if (id == null)
                {
```

```
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Клиенты клиенты = db.Клиенты.Find(id);
            if (клиенты == null)
            {
                return HttpNotFound();
            }
            return View(клиенты);
        }

        // GET: Клиенты/Create
        public ActionResult Create()
        {
             ViewBag.Номер_паспорта = new SelectList(db.Паспорта, "Номер_паспорта", "Номер_паспорта");
            return View();
        }

        public ActionResult Error()
        {
            return View();
        }

        // POST: Клиенты/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                    public ActionResult Create([Bind(Include = "Ид_клиента,Телефон,Эл_почта,Номер_паспорта")] Клиенты клиенты)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Клиенты.Add(клиенты);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
```

```
                    }
                }

                 ViewBag.Номер_паспорта = new SelectList(db.Паспорта, "Номер_паспорта", "Номер_паспорта", клиенты.Номер_паспорта);
                return View(клиенты);
            }

            // GET: Клиенты/Edit/5
            public ActionResult Edit(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Клиенты клиенты = db.Клиенты.Find(id);
                if (клиенты == null)
                {
                    return HttpNotFound();
                }
                 ViewBag.Номер_паспорта = new SelectList(db.Паспорта, "Номер_паспорта", "Номер_паспорта", клиенты.Номер_паспорта);
                return View(клиенты);
            }

            // POST: Клиенты/Edit/5
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Edit([Bind(Include = "Ид_клиента,Телефон,Эл_почта,Номер_паспорта")] Клиенты клиенты)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
                        db.Entry(клиенты).State = EntityState.Modified;
                        db.SaveChanges();
                        return RedirectToAction("Index");
                    }
                    catch
```

```
                {
                    return RedirectToAction("Error");
                }
            }
             ViewBag.Номер_паспорта = new SelectList(db.Паспорта, "Номер_паспорта", "Номер_паспорта", клиенты.Номер_паспорта);
            return View(клиенты);
        }

        // GET: Клиенты/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Клиенты клиенты = db.Клиенты.Find(id);
            if (клиенты == null)
            {
                return HttpNotFound();
            }
            return View(клиенты);
        }

        // POST: Клиенты/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Клиенты клиенты = db.Клиенты.Find(id);
            try
            {
                db.Клиенты.Remove(клиенты);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return RedirectToAction("Error");
            }

        }
```

```
            protected override void Dispose(bool disposing)
            {
                if (disposing)
                {
                    db.Dispose();
                }
                base.Dispose(disposing);
            }
        }
    }
```

НакладнаяController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;
    using System.Data.SqlClient;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class НакладнаяController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Накладная
            public ActionResult Index()
            {
                    var накладная = db.Накладная.Include(н => н.Клиенты).Include(н =>
н.Объекты_экспертизы).Include(н => н.Сотрудники);
                return View(накладная.ToList());
            }

            [HttpPost]
            public ActionResult Index(bool value, int? series, int? number)
            {
```

```
                var nakl = db.Накладная.AsEnumerable();
                if (value == true && series == null && number == null)
                {
                    if (value == true)
                        nakl = db.Накладная.Where(d => d.Статус.ToLower() == "открыт");

                    return View(nakl.ToList());
                }
                else if (value == false && series != null && number != null)
                {
                    object[] xparams = {
                    new SqlParameter("@Series", series),
                    new SqlParameter("@Nomer", number)
                    };

                        db.Database.ExecuteSqlCommand("EXEC procEditNaklad @Series, @Nomer",
xparams);

                    return View(nakl.ToList()) ;
                }
                else if (value == true)
                {
                    nakl = db.Накладная.Where(d => d.Статус.ToLower() == "открыт");

                    return View(nakl.ToList());
                }

                return View(nakl.ToList());
            }

            public ActionResult Error()
            {
                return View();
            }

                public ActionResult printFuncProgDohod(DateTime ?dateTimeFirst, DateTime ?
dateTimeSecond)
            {
                IQueryable<funcProgDohod_Result> printFunc = db.funcProgDohod(dateTimeFirst,
dateTimeSecond);

                return View(printFunc);
            }
```

```
        // GET: Накладная/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Накладная накладная = db.Накладная.Find(id);
            if (накладная == null)
            {
                return HttpNotFound();
            }
            return View(накладная);
        }

        // GET: Накладная/Create
        public ActionResult Create()
        {
            ViewBag.Ид_клиента = new SelectList(db.Клиенты, "Ид_клиента", "Ид_клиента");
               ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование");
             ViewBag.Ид_сотрудника = new SelectList(db.Сотрудники, "Ид_сотрудника", "Фа-
милия");
            return View();
        }

        // POST: Накладная/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                    public    ActionResult    Create([Bind(Include    =
"Номер,Дата_приёма,Срок_выполнения_диагностики,Причина_обращения,Состояние_объекта,Статус,При
мерная_стоимость_экспертизы,Ид_клиента,Ид_сотрудника,Ид_объекта")] Накладная накладная)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Накладная.Add(накладная);
                    db.SaveChanges();
```

```
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");

                }
            }

             ViewBag.Ид_клиента = new SelectList(db.Клиенты, "Ид_клиента", "Ид_клиента", накладная.Ид_клиента);
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта", "Наименование", накладная.Ид_объекта);
             ViewBag.Ид_сотрудника = new SelectList(db.Сотрудники, "Ид_сотрудника", "Фамилия", накладная.Ид_сотрудника);
            return View(накладная);
        }

        // GET: Накладная/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Накладная накладная = db.Накладная.Find(id);
            if (накладная == null)
            {
                return HttpNotFound();
            }
             ViewBag.Ид_клиента = new SelectList(db.Клиенты, "Ид_клиента", "Ид_клиента", накладная.Ид_клиента);
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта", "Наименование", накладная.Ид_объекта);
             ViewBag.Ид_сотрудника = new SelectList(db.Сотрудники, "Ид_сотрудника", "Фамилия", накладная.Ид_сотрудника);
            return View(накладная);
        }

        // POST: Накладная/Edit/5
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
```

```
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Edit([Bind(Include = "Номер,Дата_приёма,Срок_выполнения_диа-
гностики,Причина_обращения,Состояние_объекта,Статус,Примерная_стоимость_экспертизы,Ид_клиен-
та,Ид_сотрудника,Ид_объекта")] Накладная накладная)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
                        db.Entry(накладная).State = EntityState.Modified;
                        db.SaveChanges();
                        return RedirectToAction("Index");
                    }
                    catch
                    {

                        return RedirectToAction("Error");
                    }
                }
                 ViewBag.Ид_клиента = new SelectList(db.Клиенты, "Ид_клиента", "Ид_клиента",
накладная.Ид_клиента);
                   ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", накладная.Ид_объекта);
                 ViewBag.Ид_сотрудника = new SelectList(db.Сотрудники, "Ид_сотрудника", "Фа-
милия", накладная.Ид_сотрудника);
                return View(накладная);
            }

            // GET: Накладная/Delete/5
            public ActionResult Delete(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Накладная накладная = db.Накладная.Find(id);
                if (накладная == null)
                {
                    return HttpNotFound();
                }
                return View(накладная);
```

```
        }

        // POST: Накладная/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Накладная накладная = db.Накладная.Find(id);
            try
            {
                db.Накладная.Remove(накладная);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return RedirectToAction("Error");
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        public ActionResult IndexUser()
        {
                    var накладная = db.Накладная.Include(н => н.Клиенты).Include(н =>
н.Объекты_экспертизы).Include(н => н.Сотрудники);
            return View(накладная.ToList());
        }

        // GET: Накладная/Details/5
        public ActionResult DetailsUser(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
```

```
                Накладная накладная = db.Накладная.Find(id);
                if (накладная == null)
                {
                    return HttpNotFound();
                }
                return View(накладная);
            }
        }
    }
```

Начальная_страницаController

```
using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Web;
    using System.Web.Mvc;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class Начальная_страницаController : Controller
        {
            // GET: Начальная_страница
            public ActionResult IndexAdmin()
            {
                return View();
            }

            public ActionResult IndexUser()
            {
                return View();
            }
        }
}
```

НедвижимостьController

```
using System;
    using System.Collections.Generic;
```

```
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using DB_BureauExpertiseAndEvaluation;

namespace DB_BureauExpertiseAndEvaluation.Controllers
{
    public class НедвижимостьController : Controller
    {
        private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оценкиEntities();

        // GET: Недвижимость
        public ActionResult Index()
        {
            var недвижимость = db.Недвижимость.Include(н => н.Объекты_экспертизы);
            return View(недвижимость.ToList());
        }

        // GET: Недвижимость/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Недвижимость недвижимость = db.Недвижимость.Find(id);
            if (недвижимость == null)
            {
                return HttpNotFound();
            }
            return View(недвижимость);
        }

        public ActionResult Error()
        {
            return View();
        }

        // GET: Недвижимость/Create
```

```
        public ActionResult Create()
        {
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование");
            return View();
        }

        // POST: Недвижимость/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                        public    ActionResult    Create([Bind(Include    =
"Адрес,Район,Ид_объекта,Количество_комнат,Площадь,Тип_постройки")] Недвижимость недвижимость)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Недвижимость.Add(недвижимость);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
                }
            }

                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", недвижимость.Ид_объекта);
            return View(недвижимость);
        }

        // GET: Недвижимость/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Недвижимость недвижимость = db.Недвижимость.Find(id);
```

```
            if (недвижимость == null)
            {
                return HttpNotFound();
            }
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", недвижимость.Ид_объекта);
            return View(недвижимость);
        }

        // POST: Недвижимость/Edit/5
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "Адрес,Район,Ид_объекта,Количество_ком-
нат,Площадь,Тип_постройки")] Недвижимость недвижимость)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(недвижимость).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");
                }
            }
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", недвижимость.Ид_объекта);
            return View(недвижимость);
        }

        // GET: Недвижимость/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
```

```
            }
            Недвижимость недвижимость = db.Недвижимость.Find(id);
            if (недвижимость == null)
            {
                return HttpNotFound();
            }
            return View(недвижимость);
        }

        // POST: Недвижимость/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Недвижимость недвижимость = db.Недвижимость.Find(id);

            try
            {
                db.Недвижимость.Remove(недвижимость);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {

                return RedirectToAction("Error");
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Объекты_экспертизыController

```
using System;
    using System.Collections.Generic;
```

```
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using DB_BureauExpertiseAndEvaluation;

namespace DB_BureauExpertiseAndEvaluation.Controllers
{
    public class Объекты_экспертизыController : Controller
    {
        private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оценкиEntities();

        // GET: Объекты_экспертизы
        public ActionResult Index()
        {
            var объекты_экспертизы = db.Объекты_экспертизы.Include(о => о.Недвижимость).Include(о => о.Техника);
            return View(объекты_экспертизы.ToList());
        }

        // GET: Объекты_экспертизы/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Объекты_экспертизы объекты_экспертизы = db.Объекты_экспертизы.Find(id);
            if (объекты_экспертизы == null)
            {
                return HttpNotFound();
            }
            return View(объекты_экспертизы);
        }

        // GET: Объекты_экспертизы/Create
        public ActionResult Create()
        {
            ViewBag.Ид_объекта = new SelectList(db.Недвижимость, "Ид_объекта", "Адрес");
```

```
                ViewBag.Ид_объекта = new SelectList(db.Техника, "Ид_объекта", "Серийный_но-
мер");
                return View();
            }

            // POST: Объекты_экспертизы/Create
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
                public ActionResult Create([Bind(Include = "Ид_объекта,Наименование,Вид")]
Объекты_экспертизы объекты_экспертизы)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
                        db.Объекты_экспертизы.Add(объекты_экспертизы);
                        db.SaveChanges();
                        return RedirectToAction("Index");
                    }
                    catch
                    {
                        return RedirectToAction("Error");
                    }
                }

                ViewBag.Ид_объекта = new SelectList(db.Недвижимость, "Ид_объекта", "Адрес",
объекты_экспертизы.Ид_объекта);
                ViewBag.Ид_объекта = new SelectList(db.Техника, "Ид_объекта", "Серийный_но-
мер", объекты_экспертизы.Ид_объекта);
                return View(объекты_экспертизы);
            }

            public ActionResult Error()
            {
                return View();
            }

            // GET: Объекты_экспертизы/Edit/5
            public ActionResult Edit(int? id)
            {
```

```
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Объекты_экспертизы объекты_экспертизы = db.Объекты_экспертизы.Find(id);
                if (объекты_экспертизы == null)
                {
                    return HttpNotFound();
                }
                 ViewBag.Ид_объекта = new SelectList(db.Недвижимость, "Ид_объекта", "Адрес",
объекты_экспертизы.Ид_объекта);
                 ViewBag.Ид_объекта = new SelectList(db.Техника, "Ид_объекта", "Серийный_но-
мер", объекты_экспертизы.Ид_объекта);
                return View(объекты_экспертизы);
            }

            // POST: Объекты_экспертизы/Edit/5
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
             public ActionResult Edit([Bind(Include = "Ид_объекта,Наименование,Вид")] Объек-
ты_экспертизы объекты_экспертизы)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
                        db.Entry(объекты_экспертизы).State = EntityState.Modified;
                        db.SaveChanges();
                        return RedirectToAction("Index");
                    }
                    catch
                    {

                        return RedirectToAction("Error");
                    }
                }
                 ViewBag.Ид_объекта = new SelectList(db.Недвижимость, "Ид_объекта", "Адрес",
объекты_экспертизы.Ид_объекта);
                 ViewBag.Ид_объекта = new SelectList(db.Техника, "Ид_объекта", "Серийный_но-
мер", объекты_экспертизы.Ид_объекта);
```

```
            return View(объекты_экспертизы);
        }

        // GET: Объекты_экспертизы/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Объекты_экспертизы объекты_экспертизы = db.Объекты_экспертизы.Find(id);
            if (объекты_экспертизы == null)
            {
                return HttpNotFound();
            }
            return View(объекты_экспертизы);
        }

        // POST: Объекты_экспертизы/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Объекты_экспертизы объекты_экспертизы = db.Объекты_экспертизы.Find(id);

            try
            {
                db.Объекты_экспертизы.Remove(объекты_экспертизы);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {

                return RedirectToAction("Error");
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
```

```
                }
                base.Dispose(disposing);
            }
        }
    }
```

ОтделController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class ОтделController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Отдел
            public ActionResult Index()
            {
                return View(db.Отдел.ToList());
            }

            // GET: Отдел/Details/5
            public ActionResult Details(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Отдел отдел = db.Отдел.Find(id);
                if (отдел == null)
                {
                    return HttpNotFound();
                }
                return View(отдел);
```

```
        }

        // GET: Отдел/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Отдел/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
          public ActionResult Create([Bind(Include = "Ид_отдела,Наименование_отдела,Начальник_отдела,Телефон,Зам_начальника")] Отдел отдел)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Отдел.Add(отдел);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");
                }

            }

            return View(отдел);
        }

        // GET: Отдел/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
```

```
            Отдел отдел = db.Отдел.Find(id);
            if (отдел == null)
            {
                return HttpNotFound();
            }
            return View(отдел);
        }

        // POST: Отдел/Edit/5
         // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
         public ActionResult Edit([Bind(Include = "Ид_отдела,Наименование_отдела,Начальник_отдела,Телефон,Зам_начальника")] Отдел отдел)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(отдел).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");
                }
            }
            return View(отдел);
        }

        public ActionResult Error()
        {
            return View();
        }
        // GET: Отдел/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
```

```
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Отдел отдел = db.Отдел.Find(id);
            if (отдел == null)
            {
                return HttpNotFound();
            }
            return View(отдел);
        }

        // POST: Отдел/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Отдел отдел = db.Отдел.Find(id);
            try
            {
                db.Отдел.Remove(отдел);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return RedirectToAction("Error");
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

ПаспортаController

```
using System;
    using System.Collections.Generic;
    using System.Data;
```

```
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using DB_BureauExpertiseAndEvaluation;

namespace DB_BureauExpertiseAndEvaluation.Controllers
{
    public class ПаспортаController : Controller
    {
        private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оценкиEntities();

        // GET: Паспорта
        public ActionResult Index()
        {
            return View(db.Паспорта.ToList());
        }

        /// Поиск клиентов в сущности Паспорта
        [HttpPost]
        public ActionResult Index(string name, string surname)
        {
            var passport = db.Паспорта.AsEnumerable();
            if (!String.IsNullOrEmpty(name) && !String.IsNullOrEmpty(surname))
            {
                passport = db.Паспорта.Where(n => n.Имя == name).Where(s => s.Фамилия == surname).Select(k => k);
            }
            else
            {
                passport = db.Паспорта;
            }
            return View(passport.ToList());
        }

        // GET: Паспорта/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
```

```
            }
            Паспорта паспорта = db.Паспорта.Find(id);
            if (паспорта == null)
            {
                return HttpNotFound();
            }
            return View(паспорта);
        }

        // GET: Паспорта/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Паспорта/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                    public ActionResult Create([Bind(Include = "Номер_паспорта,Серия_паспорта,Дата_выдачи,Дата_окончания,Адрес,Фамилия,Имя,Отчество,Дата_рождения,Пол")] Паспорта паспорта)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Паспорта.Add(паспорта);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");
                }
            }

            return View(паспорта);
        }
```

```
        // GET: Паспорта/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Паспорта паспорта = db.Паспорта.Find(id);
            if (паспорта == null)
            {
                return HttpNotFound();
            }
            return View(паспорта);
        }

        // POST: Паспорта/Edit/5
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
         public ActionResult Edit([Bind(Include = "Номер_паспорта,Серия_паспорта,Дата_выдачи,Дата_окончания,Адрес,Фамилия,Имя,Отчество,Дата_рождения,Пол")] Паспорта паспорта)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(паспорта).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch (Exception)
                {

                    return RedirectToAction("Error");
                }
            }
            return View(паспорта);
        }

        // GET: Паспорта/Delete/5
        public ActionResult Delete(int? id)
```

```
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Паспорта паспорта = db.Паспорта.Find(id);
    if (паспорта == null)
    {
        return HttpNotFound();
    }
    return View(паспорта);
}

public ActionResult Error()
{
    return View();
}

// POST: Паспорта/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Паспорта паспорта = db.Паспорта.Find(id);
    try
    {
        db.Паспорта.Remove(паспорта);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    catch
    {
        return RedirectToAction("Error");
    }
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
```

```
            }
        }
    }
```

СотрудникиController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;
    using System.Data.SqlClient;
    using System.Diagnostics;
    using DB_BureauExpertiseAndEvaluation.Models;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class СотрудникиController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Сотрудники
            public ActionResult Index()
            {

                var сотрудники = db.Сотрудники.Include(с => с.Отдел);
                return View(сотрудники.ToList());
            }

            /// ПОИСК И ФИЛЬТРАЦИЯ
            [HttpPost]
            public ActionResult Index(string text, string name, string surname)
            {

                var sotr = db.Сотрудники.AsEnumerable();

                if (text != "Все")
                {
```

```
                    if (!String.IsNullOrEmpty(text))
                    {
                            sotr = db.Сотрудники.Where(n => n.Отдел.Наименование_отдела ==
text).ToList();
                    }

                    return View(sotr.ToList());
                }
                else if (name != null && surname != null && text == "Все")
                {

                    if (!String.IsNullOrEmpty(name) && !String.IsNullOrEmpty(surname))
                    {
                         sotr = db.Сотрудники.Where(n => n.Имя == name).Where(s => s.Фамилия
== surname).Select(k => k);
                    }
                    return View(sotr.ToList());
                }
                else
                {

                        sotr = db.Сотрудники.Where(n => n.Отдел.Наименование_отдела ==
text).ToList().Where(n => n.Имя == name).Where(s => s.Фамилия == surname).Select(k => k);
                    return View(sotr.ToList());
                }

                return View(sotr.ToList());
            }

            /// ФУНКЦИЯ
            public ActionResult printFuncZarplata(int ?Id)
            {
                IQueryable<funcZarplata_ResultF> printFunc = db.funcZarplata(Id);

                return View(printFunc);
            }

            // GET: Сотрудники/Details/5
            public ActionResult Details(int? id)
            {
                if (id == null)
                {
```

```
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (сотрудники == null)
            {
                return HttpNotFound();
            }
            return View(сотрудники);
        }

        // GET: Сотрудники/Create
        public ActionResult Create()
        {
                        ViewBag.Ид_отдела = new SelectList(db.Отдел, "Ид_отдела",
"Наименование_отдела");
            return View();
        }

        // POST: Сотрудники/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                public ActionResult Create([Bind(Include =
"Ид_сотрудника,Фамилия,Имя,Отчество,Дата_рождения,Адрес_проживания,Стаж,Телефон,Номер_трудо-
вой_книжки,ИНН,Зарплата,Серия_паспорта,Номер_паспорта,Ид_отдела")] Сотрудники сотрудники)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Сотрудники.Add(сотрудники);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
                }
            }
```

```
                                ViewBag.Ид_отдела = new SelectList(db.Отдел, "Ид_отдела",
"Наименование_отдела", сотрудники.Ид_отдела);
                return View(сотрудники);
            }

            public ActionResult Error()
            {
                return View();
            }

            // GET: Сотрудники/Edit/5
            public ActionResult Edit(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Сотрудники сотрудники = db.Сотрудники.Find(id);
                if (сотрудники == null)
                {
                    return HttpNotFound();
                }
                                ViewBag.Ид_отдела = new SelectList(db.Отдел, "Ид_отдела",
"Наименование_отдела", сотрудники.Ид_отдела);
                return View(сотрудники);
            }

            // POST: Сотрудники/Edit/5
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
                                        public ActionResult Edit([Bind(Include =
"Ид_сотрудника,Фамилия,Имя,Отчество,Дата_рождения,Адрес_проживания,Стаж,Телефон,Номер_трудо-
вой_книжки,ИНН,Зарплата,Серия_паспорта,Номер_паспорта,Ид_отдела")] Сотрудники сотрудники)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
                        db.Entry(сотрудники).State = EntityState.Modified;
                        db.SaveChanges();
```

```
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
                }
            }
                        ViewBag.Ид_отдела = new SelectList(db.Отдел, "Ид_отдела",
"Наименование_отдела", сотрудники.Ид_отдела);
            return View(сотрудники);
        }

        // GET: Сотрудники/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (сотрудники == null)
            {
                return HttpNotFound();
            }
            return View(сотрудники);
        }

        // POST: Сотрудники/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Сотрудники сотрудники = db.Сотрудники.Find(id);

            try
            {
                db.Сотрудники.Remove(сотрудники);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return RedirectToAction("Error");
```

```
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

ТехникаController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class ТехникаController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Техника
            public ActionResult Index()
            {
                var техника = db.Техника.Include(т => т.Объекты_экспертизы);
                return View(техника.ToList());
            }

            // GET: Техника/Details/5
            public ActionResult Details(int? id)
            {
```

```
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Техника техника = db.Техника.Find(id);
            if (техника == null)
            {
                return HttpNotFound();
            }
            return View(техника);
        }

        // GET: Техника/Create
        public ActionResult Create()
        {
               ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование");
            return View();
        }

        public ActionResult Error()
        {
            return View();
        }

        // POST: Техника/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
                                    public    ActionResult    Create([Bind(Include    =
"Год_производства,Серийный_номер,Цена_при_покупке,Ид_объекта,Компания_производитель,Модель")]
Техника техника)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Техника.Add(техника);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
```

```
                    catch
                    {
                        return RedirectToAction("Error");
                    }

                }

                    ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", техника.Ид_объекта);
                return View(техника);
            }

            // GET: Техника/Edit/5
            public ActionResult Edit(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Техника техника = db.Техника.Find(id);
                if (техника == null)
                {
                    return HttpNotFound();
                }
                    ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", техника.Ид_объекта);
                return View(техника);
            }

            // POST: Техника/Edit/5
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
                                        public     ActionResult     Edit([Bind(Include     =
"Год_производства,Серийный_номер,Цена_при_покупке,Ид_объекта,Компания_производитель,Модель")]
Техника техника)
            {
                if (ModelState.IsValid)
                {
                    try
                    {
```

```
                    db.Entry(техника).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    return RedirectToAction("Error");
                }

            }
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", техника.Ид_объекта);
            return View(техника);
        }

        // GET: Техника/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Техника техника = db.Техника.Find(id);
            if (техника == null)
            {
                return HttpNotFound();
            }
            return View(техника);
        }

        // POST: Техника/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Техника техника = db.Техника.Find(id);
            try
            {
                db.Техника.Remove(техника);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
```

```
                {

                    return RedirectToAction("Error");
                }


            }

            protected override void Dispose(bool disposing)
            {
                if (disposing)
                {
                    db.Dispose();
                }
                base.Dispose(disposing);
            }
        }
    }
```

Экспертиза_и_оценкаController

```
using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.Entity;
    using System.Linq;
    using System.Net;
    using System.Web;
    using System.Web.Mvc;
    using DB_BureauExpertiseAndEvaluation;

    namespace DB_BureauExpertiseAndEvaluation.Controllers
    {
        public class Экспертиза_и_оценкаController : Controller
        {
                private Бюро_экспертизы_и_оценкиEntities db = new Бюро_экспертизы_и_оцен-
киEntities();

            // GET: Экспертиза_и_оценка
            public ActionResult Index()
            {
                var экспертиза_и_оценка = db.Экспертиза_и_оценка.Include(э => э.Объекты_экс-
пертизы);
```

```
            return View(экспертиза_и_оценка.ToList());
        }

        // GET: Экспертиза_и_оценка/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Экспертиза_и_оценка экспертиза_и_оценка = db.Экспертиза_и_оценка.Find(id);
            if (экспертиза_и_оценка == null)
            {
                return HttpNotFound();
            }
            return View(экспертиза_и_оценка);
        }

        // GET: Экспертиза_и_оценка/Create
        public ActionResult Create()
        {
                ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование");
            return View();
        }


        public ActionResult Error()
        {
            return View();
        }

        // POST: Экспертиза_и_оценка/Create
          // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
          public ActionResult Create([Bind(Include = "Ид_диагностики,Ид_объекта,Вид_диа-
гностики,Количество_дней")] Экспертиза_и_оценка экспертиза_и_оценка)
        {
            if (ModelState.IsValid)
            {
```

```
                    try
                    {
                        db.Экспертиза_и_оценка.Add(экспертиза_и_оценка);
                        db.SaveChanges();
                        return RedirectToAction("Index");
                    }
                    catch
                    {

                        return RedirectToAction("Error");
                    }

                }

                    ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", экспертиза_и_оценка.Ид_объекта);
                return View(экспертиза_и_оценка);
            }

            // GET: Экспертиза_и_оценка/Edit/5
            public ActionResult Edit(int? id)
            {
                if (id == null)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
                Экспертиза_и_оценка экспертиза_и_оценка = db.Экспертиза_и_оценка.Find(id);
                if (экспертиза_и_оценка == null)
                {
                    return HttpNotFound();
                }
                    ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", экспертиза_и_оценка.Ид_объекта);
                return View(экспертиза_и_оценка);
            }

            // POST: Экспертиза_и_оценка/Edit/5
              // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
            // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
            [HttpPost]
            [ValidateAntiForgeryToken]
```

```
        public ActionResult Edit([Bind(Include = "Ид_диагностики,Ид_объекта,Вид_диагно-
стики,Количество_дней")] Экспертиза_и_оценка экспертиза_и_оценка)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(экспертиза_и_оценка).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {

                    return RedirectToAction("Error");
                }
            }
               ViewBag.Ид_объекта = new SelectList(db.Объекты_экспертизы, "Ид_объекта",
"Наименование", экспертиза_и_оценка.Ид_объекта);
            return View(экспертиза_и_оценка);
        }

        // GET: Экспертиза_и_оценка/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Экспертиза_и_оценка экспертиза_и_оценка = db.Экспертиза_и_оценка.Find(id);
            if (экспертиза_и_оценка == null)
            {
                return HttpNotFound();
            }
            return View(экспертиза_и_оценка);
        }

        // POST: Экспертиза_и_оценка/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Экспертиза_и_оценка экспертиза_и_оценка = db.Экспертиза_и_оценка.Find(id);
```

```
            try
            {
                db.Экспертиза_и_оценка.Remove(экспертиза_и_оценка);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {

                return RedirectToAction("Error");
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```