**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 студии звукозаписи*

Выполнил студент  */ Кучина А. А. /*

*подпись инициалы, фамилия*

Курс  *2* Группа  *ИСТбд-22*

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

20 .

**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*

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

студенту  *ИСТбд-22 Кучина А. А.*

*группа фамилия, инициалы*

Тема работы  *Автоматизированная информационная система студии звукозаписи*

Срок сдачи законченной работы « » 20 г.

Исходные данные к работе *методические указания к выполнению курсовой*

*(базовое предприятие, характер курсовой работы:*

*работы и проведению практических занятий для студентов направления*

*задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)*

*09.03.03 «Информационные системы и технологии» по дисциплине*

*«Базы данных» Родионова В.В.*

Содержание пояснительной записки *список использованных обозначений*

*и сокращений, 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  */ Кучина А. А. /*

*подпись инициалы, фамилия*

« » 20 г.

# Содержание

[Введение 4](#_Toc74926352)

[1 Техническое задание 7](#_Toc74926353)

[1.1 Общие сведения 7](#_Toc74926354)

[1.2 Назначение и цели создания системы 7](#_Toc74926355)

[1.3 Характеристика объекта автоматизации 7](#_Toc74926358)

[1.4 Требования к системе 8](#_Toc74926359)

[1.5 Состав и содержание работ по созданию системы 10](#_Toc74926363)

[1.6 Порядок контроля и приёмки системы 10](#_Toc74926364)

[1.7 Требования к документированию 10](#_Toc74926365)

[2 Информационное обеспечение системы 11](#_Toc74926366)

[2.1 Концептуальная схема базы данных 11](#_Toc74926367)

[2.2 Внутренняя схема базы данных 18](#_Toc74926371)

[3 Алгоритмическое обеспечение системы 22](#_Toc74926373)

[3.1 Общая характеристика 23](#_Toc74926374)

[3.2. Используемые данные 23](#_Toc74926375)

[3.3. Результаты выполнения 23](#_Toc74926376)

[3.4 Логическое описание 24](#_Toc74926377)

[4 Прикладное программное обеспечение системы 24](#_Toc74926379)

[4.1 Общая характеристика прикладного программного обеспечения 25](#_Toc74926380)

[4.2 Структура и состав прикладного программного обеспечения 26](#_Toc74926381)

[4.3 Особенности реализации и сопровождения 29](#_Toc74926382)

[5 Руководство пользователя 30](#_Toc74926383)

[5.1 Общие сведения 30](#_Toc74926384)

[5.2 Порядок и особенности работы 31](#_Toc74926385)

[5.3 Исключительные ситуации 38](#_Toc74926386)

[Список использованных источников 40](#_Toc74926387)

[Приложение А. Исходные тексты программных модулей 42](#_Toc74926388)

# Введение

Сегодня не малое количество людей хотело бы записать свою песню. Многие создают музыкальные композиции прямо дома, записывая все части на обычный микрофон и сводя дорожки в бесплатных программах. Это, безусловно, самый дешевый способ создать песню, однако домашняя запись не отличается особым качеством. Для получения действительно качественной композиции стоит обратиться к профессионалам – в студию звукозаписи.

Самым важным человеком в студии звукозаписи является звукорежиссер. Он и есть тот самый «волшебник», который превращает набор звуков в законченную композицию со своей изюминкой. Звукорежиссер должен иметь прекрасный мзыкальный слух, разбираться в жанрах. Это человек, для которого музыка становится полотном, набором звуков, изменяя которые можно создавать абсолютно разное настроение, задавать композиции разную стилистику. Как правило, если студия небольшая и имеет небольшое количество заказов, то справиться звукорежиссер может и в одиноччку. Он будет ремонтировать обордование, заключать договора с клиентами и работать непосредственно с песнями. Звукорежиссер может выполнять обязанности аранжировщика, менеджера, администратора, техобслуживания. Однако в более крупных студиях для кождой должости нанимается отдельный человек, иногда даже не один.

Аранжировщик – человек, выявляющий изюминку композиции. Он координирует исполнителя по жанрам, подбирает нужный темп, скорость, тональность композиции.

Однако просто записать песню недостаточно, чтобы заявить себя исполнителем и найти слушателей и поклонников. Для того, чтобы добиться успеха исполнителю необходима координация в музыкальном мире, наведение на определенные современные тренды. Этим занимается музыкальный продюссер. Он продвигает исполнителя в топы, занимается развитием его «музыкального лица». Обычно музыкальный продюссер сам ищет талантливых людей, однако, если студия и продюссер уже широко известны массам, таланты ищут таких профессионалов самостоятельно. Работа продюссера не менее ценна, чем работа звукорежиссера, ведь если последний отвечает за техническую сторону музыкальной композиции, за ее качество, то режиссер отвечает за развитие самого исполнителя.

Конечно же в студии звукозаписи производится обслуживание техники. И если этим не занимается звукорежиссер, то эту обязанность выполняют системный администратор и инженер (техник). Системный администратор занимается настройкой всех программ для их лучшей синхронизации между собой и между другим оборудованием (микрофоны, наушники, звукоуселители, гитары и прочие музыкальные инструменты). А вот инженер следит за «железной» частью всей аппаратуры. В его обязанности входит поддержание работоспособности технических устройств, их своевременных ремонт и своевременная проверка для профилактики сбоев.

Помещения в студии звукозаписи так же имеют специальное оборудование и главное – хорошую звукоизоляцию для получения чистой записи, с которой в последствии будет удобно и приятно работать. Оборудование же включает в себя несколько музыкальных инструментов, акустическую систему, мощный компьютер для работы аранжировщика и для сведения дорожек в единую композицию, несколько различных микрофонов для записи голосов (основноко и бэквокала) и инструментов, DAW контроллер, предусилитель, отдельную звуковую карту (поскольку та, что встроенна в компьютер не пригодна для профессиональной работы), микшерный пульт для настройки баланса прямо во время записи.

При анализе предметной области были использованы следующие ресурсы:

1. <http://sintez-audio.ru/news/117-2015-10-23-07-26-27> - Сведения о работе звукорежиссера, об основном внутреннем устройстве студии звукозаписи и процессе создания музыкальной композиции.
2. <https://arefyevstudio.com/2017/04/28/kakoe-oborudovanie-nuzhno-dlya-studii-zvukozapisi/> - данная статья раскрыла тему технического оснащения звукозаписывающей студии.
3. <http://sintez-audio.ru/news/125-2015-11-23-06-44-25> - приоткрыта завеса над профессией аранжировщика и его обязанностями
4. <http://sintez-audio.ru/news/126-2015-12-09-07-12-31> - в данной статье представлены сведения о работе технического персонала студии звукозаписи.

При непосредственном создании продукта были использованы следующие источники:

1. <https://smarly.net/pro-asp-net-mvc-4> - на данном ресуре были подчерпнуты основные принципы работы с проектом ASP.NET MVC, концепции создания приложения с использованием MVC Framework.
2. <https://metanit.com/sharp/mvc5/> - так же руководство по работе с ASP.NET MVC, только уже последней пятой версии.
3. Методические указания к лабораторной работе №2 – здесь были почерпнуты необходимые знания для создания хранимых процедур и определяемых пользователем функций
4. Методические указания к лабораторной работе №3 – из этого источника была подчерпнута информация о точечной нотации LINQ, необходимой для определения функций фильтрации и поиска. Также данные методические указания предоставляют базовую информацию о понятии лямбда-выражение и интерфейсе IEnumerable.

# 1 Техническое задание

## 1.1 Общие сведения

Автоматизированная информационная система студии звукозаписи, далее система.

## 1.2 Назначение и цели создания системы

### 1.2.1 Назначение системы

Система представляет собой удобный способ хранения данных, необходимых студии звукозаписи для нормального функционирования и работы с клиентами. Данная система может быть доступна студиям звукозаписи и в некоторых случаях, когда пользователь создает «домашнюю» студию звукозаписи.

### 1.2.2 Цели создания системы

* Удобство получения информации об услугах, выполняемых над конкретной песней
* Систематизированное хранение информации о заключенных со студией звукозаписи договорах
* Систематизированное хранение данных персонала студии звукозаписи
* Организация каталога услуг, которые предоставляет студия звукозаписи, а также списка выполненных и выполняемых услуг в виде перечня с указанием количества рабочих дней, потраченных на выполнение услуги и конечную стоимость.

## 1.3 Характеристика объекта автоматизации

Студия звукозаписи представляет собой организацию, предоставляющую услуги по записи песен и альбомов, обработке музыкальных композиций. Звукорежиссер следит за качеством записываемой композиции. Это человек, который так же старается максимально сильно передать особую атмосферу через записанную композицию. Направление работы звукоинженера касается сведения и мастеринга. Человек этой профессии работает с различными звукообрабатывающими программами, добавляет эффекты, доводит композицию до профессионального уровня.

Весь цикл появления песни выглядит следующим образом. Сначала заказчик (он же исполнитель) пишет свой текст или музыку. С этим материалом он приходит в студию, где сначала производится запись голоса, если он присутствует, инструментов. Далее мастеринг-инженер редактирует полученные записи, поправляет громкость, применяет автотюн – программа, которая позволяет подкорректировать звук, делая ноты более чистыми, применяет различные фильтры, придавая композиции уникальность. Затем мастер сведения соответственно сводит все дорожки, убирая накладки по частоте звука. В итоге получается цельная музыкальная композиция.

В студии можно записать не просто одну песню, называемую синглом, а целый альбом песен. Существует особый вид альбомов – ЕР, что значит мини-альбом от английского extended play. В таких мини-альбомах содержится до четырех музыкальных композиций, поскольку раньше, когда песни записывались на грам-пластинки, на них умещалось как раз таки не более четырех песен. Сейчас такой формат альбомов все еще остается.

Работа ведется как с сольными исполнителями, так и с музыкальными группами. При ремастеринге какой-то из своих композиций, исполнителю может не понадобиться заново записывать сольное исполнение и остальные части композиции. Он может прийти с уже имеющимися записями.

## 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 он несанкционированного доступа

Неаутентифицированный пользователь может просматривать информацию только о Песнях, записанных в студии звукозаписи (название песни, альбом, длина песни), Исполнителях, с которыми работала или работает студия (его имя, количество записанных альбомов, дату начала карьеры, количество наград), и альбомах, записанных на студии (Название, жанр, общую длину альбома, количество песен в нем, язык пеен и имя исполнителя). Модификация данных для такого пользователя запрещена.

В системе определяются три различных типа пользователей.

1. Неаутентифицированный пользователь (далее Гость). В системе нет реализации регистрации и авторизации, поэтому существует возможность просмотра некоторых таблиц для тех пользователей, которые не являются сотрудниками студии звукозаписи, поскольку личного аккаунта у них нет. Они располагают доступам к минимальному набору информации, указанному выше.
2. Аутентифицированный пользователь с ограниченными функциями просмотра таблиц (далее Пользователь). Пользователю доступны те же таблицы, что и Гостю с возможностью просматривать данные о стране Исполнителя. Также Пользователю доступна информация о сотрудниках студии звукозаписи: их имя и должность. Пользователь может выполнять поиск и фильтрацию при наличии оных для конкретной таблицы, которая ему доступна.
3. Аутентифицированный пользователь, использующий все возможности системы (далее Администратор). Как следует из определения данного типа пользователя системы, он может неограниченно пользоваться функционалом системы.

В базе данных находится специальная таблица с логинами, паролями и ролями для всех сотрудников студии звукозаписи.

### 1.4.2 Требования к функциям, выполняемым системой

1. Хранение информации о сотрудниках студии звукозаписи
2. Учет песен, записанных в студии звукозаписи
3. Хранение информации об альбомах, содержащих песни
4. Хранение информации о клиентах студии звукозаписи
5. Хранение данных о договорах, заключенных с исполнителями песен

### 1.4.3 Требования к видам обеспечения

1.4.3.1 Требования к техническому обеспечению

Рекомендуемая конфигурации технического обеспечения:

* Процессор: Intel(R) Pentium(R) CPU 4405U @ 2.10GHz, 2101 МГц, ядер: 2, логических процессоров: 4;
* Количество ядер: 2;
* Материнская плата: T-Rex\_SK;
* Жесткий диск: Samsung SSD 860 EVO 250GB;
* Видеокарта: Intel® HD Graphics 510;
* Оперативная память: 8,00 ГБ;
* Операционная система: Windows 10 Домашняя для одного языка;
* Звуковая карта: Realtek High Definition Audio.

1.4.3.2 Требования к программному обеспечению

* Операционная система: Windows 10 Домашняя для одного языка;
* Инструментальная среда разработки: Visual Studio 2019;
* СУБД: Microsoft SQL Management Server 2019;
* ПО, использованное при создании модели «сущность-связь»: ER-constructor

## 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:

1. Анализ концептуальной схемы

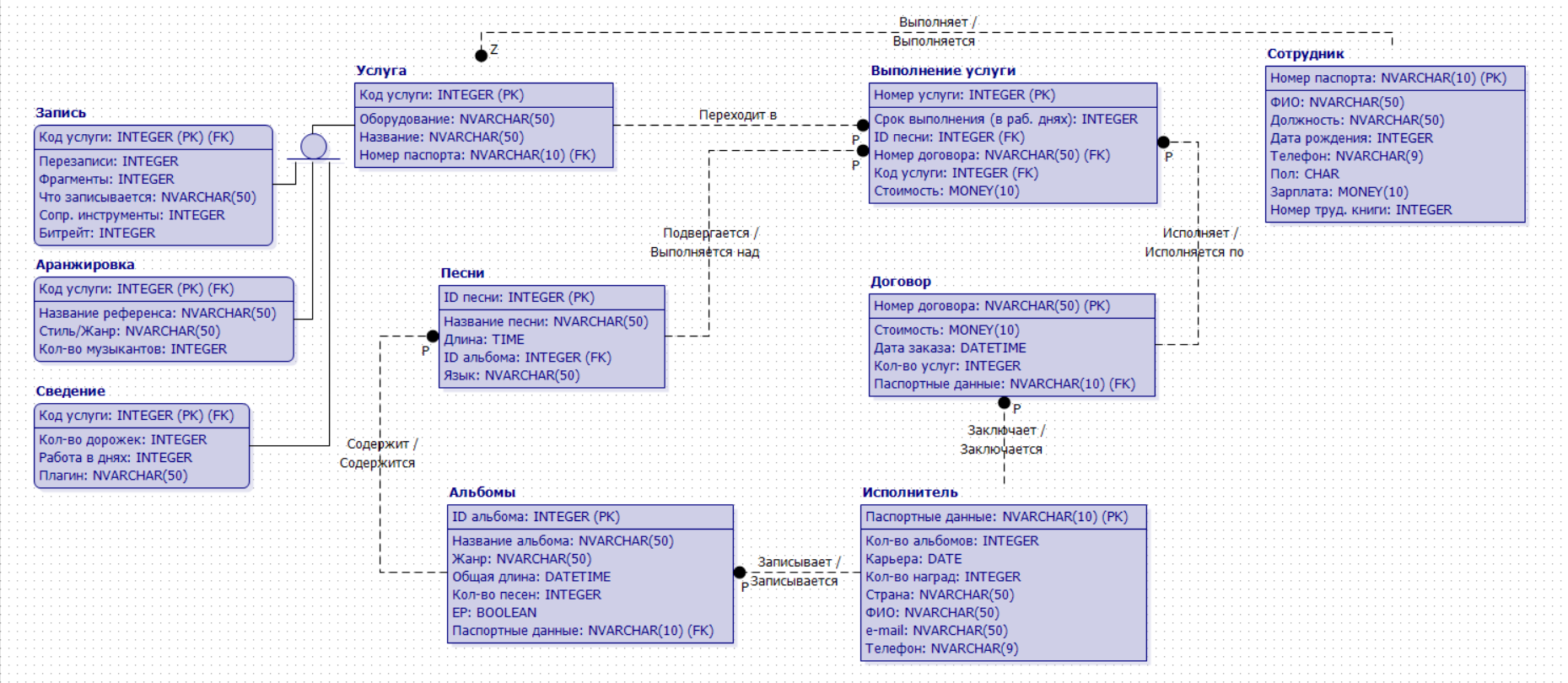
2. Список использованных обозначений и сокращений

3. Заключение.

# 2 Информационное обеспечение системы

## 2.1 Концептуальная схема базы данных

### 2.1.1 Модель «сущность-связь»



### 2.1.2 Сущности и их атрибуты

|  |  |
| --- | --- |
| **Название сущности** | **Описание сущности** |
| Сотрудник | Сотрудники, работающие в студии звукозаписи. Идентифицируются по паспортным данным, содержащим серию и номер паспорта. Атрибут «Код услуги» отвечает за тип услуги, выполняемой данным сотрудником. |
| Договор | Трудовой договор, заключающийся между клиентом(исполнителем) и студией звукозаписи. Идентифицируется по особому сложному номеру. В нем содержатся паспортные данные клиента (исполнителя) и количество услуг по заказу клиента. |
| Исполнитель | Исполнители, заключающие договор со студией звукозаписи. Они же считаются клиентами студии. Идентифицируются по паспортным данным по тому же принципу, что и сотрудники. Атрибут «карьера» отвечает за официальную дату начала карьеры в качестве музыкального исполнителя. Атрибут «награды» отвечает за все награды, полученные исполнителем за время его музыкальной карьеры. |
| Альбомы | Альбомы, записанные в студии звукозаписи. Идентифицируются по простому номеру. Атрибут «общая длина» обозначает общую длительность альбома по времени в минутах и секундах. Атрибут «ЕР» говорит о том, является ли альбом «мини-альбомом». В данные об альбоме заносятся паспортные данные его исполнителя. |
| Песни | Песни, записанные в студии звукозаписи. Идентифицируются по простому номеру. Атрибут «Длина» отвечает за длину трека в минутах и секундах. Атрибут «Язык» отвечает за язык, на котором написан текст песни. |
| Услуга | Список возможных услуг. Идентифицируются по особому коду услуг. Атрибут «Оборудование» отвечает за оборудование, которое задействовано в процессе выполнения услуги. |
| Выполнение услуги | Сводка данных по всем выполненным услугам со сроком выполнения этой услуги в рабочих днях, идентификатором песни, над которой производилась данная услуга, а так же конечной стоимостью услуги и номером договора, к которому эта услуга относится. |
| Запись | Подтип сущности Услуга. Представляет собой тип услуги – запись песни с указанием количества произведенных перезаписей трека (атрибут «Перезаписи»), количества фрагментов записи (атрибут «Фрагменты»). Атрибут «что записывается» отвечает за записываемый исполнителем инструмент или его голос. Сопровождающие инструменты – вспомогательные музыкальные инструменты, использованные для создания трека. Битрейт – это своеобразная мера качества записи, выражаемая в количестве бит, пришедшихся на секунду трека |
| Аранжировка | Подтип сущности Услуга. Представляет собой тип услуги – аранжировка имеющегося фрагмента песни с указанием определенных референсов (атрибут «Референсы») для создания нужной атмосферы в песне. Референсы, если они есть, предоставляются заказчиком заранее. Атрибут «количество музыкантов» отвечает за количество музыкантов, дополнительно привлеченных для выполнения услуги |
| Сведение | Подтип сущности Услуга. Представляет собой тип услуги – сведение фрагментов песни в единый трек. Фрагменты могут быть записаны исполнителем прямо в студии звукозаписи, либо же предоставлены заранее. Указывается количество дорожек, которые необходимо соединить в трек и плагин – дополнительные библиотеки, используемые при сведении дорожек |

### 2.1.3 Связи между сущностями

|  |  |
| --- | --- |
| **Название связи** | **Описание связи** |
| Выполняет/Выполняется | Неидентифицирующая связь принадлежности между сущностями Сотрудник (родитель) и Услуга (потомок). Услуга выполняется сотрудником, сотрудник выполняет услугу. Тип связи «один к одному». Минимальное кардинальное число для сущности-родителя равно 0, поскольку сотрудник не обязательно выполняет услугу, представленную подтипом сущности-потомка (например, менеджер, бухгалтер – они не занимаются работой непосредственно с песнями). Максимальное кардинальное число для сущности- потомка равно 1, поскольку услуга является своеобразной профессиональной областью сотрудника. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку услуга должна обязательно выполняться сотрудником, притом только одним. |
| Исполняет/Исполняется по | Неидентифицирующая связь принадлежности между сущностями Договор (родитель) и Выполнение услуги (потомок). Услуга исполняется по договору, и договор приводит услугу в выполнение. Тип связи «Один ко многим». Минимальное кардинальное число для сущности-родителя равно 1, поскольку в договоре должна быть указана хотя бы одна услуга. Максимальное кардинальное число для сушности-родителя равно N, поскольку в договоре может указываться несколько услуг. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку каждая услуга должна быть обязательно указана в каком-либо договоре, причем только в одном. |
| Заключает/Заключается | Неидентифицирующая связь принадлежности между сущностями Исполнитель (родитель) и Договор (потомок). Исполнитель заключает договор, и договор заключается исполнителем. Тип связи «один ко многим». Минимальное кардинальное число для сущности-родителя равно 1, поскольку не имеет смысла хранить в базе данных информацию об исполнителях, не заключавших договор со студией звукозаписи. Максимальное кардинальное число для сущности-родителя равно N, поскольку исполнитель может заключать неограниченное количество договоров со студией. Минимальное и максимальное кардинальное число для сущности-потомка равны 1, поскольку договор может заключить только один исполнитель, не больше и не меньше. |
| Записывает/Записывается | Неидентифицирующая связь принадлежности между сущностями Исполнитель (родитель) и Альбомы (потомок). Альбом записывается исполнителем, и исполнитель записывает альбом. Тип связи «один ко многим». Минимальное кардинальное число для сущности-родителя равно 1, поскольку не имеет смысла хранить данные об исполнителе, который не записал ни одного альбома, включая альбомы с одной единственной песней (альбом-сингл). Максимальное кардинальное число для сущности-родителя равно N, поскольку альбомов может быть записано неограниченное количество. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку один альбом может быть записан только одним исполнителем. |
| Содержит/Содержится | Неидентифицирующая связь принадлежности между сущностями Альбомы (родитель) и Песни (потомок). Альбом содержит песни, и песни содержатся в альбоме. Тип связи «один ко многим». Минимальное кардинальное число для сущности-родителя равно 1, поскольку в альбоме должна быть хотя бы одна песня. Максимальное кардинальное число для сущности-родителя равно N, поскольку в альбоме может быть неограниченное число песен. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку песня обязательно должна входить в один альбом. |
| Подвергается/Выполняется над | Неидентифицирующая связь принадлежности между сущностями Песни (родитель) и Выполнение услуг (потомок). Тип связи «Один ко многим». Минимальное кардинальное число для сущности-родителя равно 1, поскольку для выполнения услуги нужна запись о песне, над которой будет выполняться услуга. Максимальное кардинальное число для сущности-родителя равно N, поскольку над песней может совершаться неограниченное число услуг. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку выполнение одной услуги в сводке услуг возможно исключительно над одной песней и невозможно при отсутствии песни в принципе. |
| Переходит в | Неидентифицирующая связь принадлежности между сущностями Услуга(родитель) и Выполнение услуги (потомок). Тип связи «один ко многим». Здесь стоит уточнить, что сущность Услуга является неполным перечнем предоставляемых студией звукозаписи услуг, а сущность Выполнение услуги – сводка данных по всем выполненным услугам над конкретными песнями. Поэтому минимальное кардинальное число для сущности-родителя равно 1, поскольку каждая услуга должна быть хоть раз выполнена. Максимальное кардинальное число для сущности-родителя равно N, поскольку одна услуга может фигурировать в различных песнях, а значит, может выполняться неоднократно для разных песен. Минимальное и максимальное кардинальные числа для сущности-потомка равны 1, поскольку для того, чтобы выполнить какую-либо услугу для конкретной песни, она должна присутствовать в Перечне услуг, причем выполняется только одна услуга из Перечня. |
| Дискриминант | Связь типа «Имеет» между сущностью Услуга и подсущностями «Запись», «Аранжировка» и «Сведение». Данная связь описывает неполный категориальный кластер. Кластер является неполным, поскольку студия звукозаписи может предоставлять услуги не только Записи, Аранжировки и Сведения, но в данной модели рассматриваются именно эти услуги. Минимальные категориальные числа со стороны потомков равны 1. |

## 2.2 Внутренняя схема базы данных

### 2.2.1 База данных системы

2.2.1.1 Скрипты создания таблиц

1. Таблица аутентификации

CREATE TABLE [dbo].[Autentification](

[Логин] [nvarchar](10) NOT NULL,

[Пароль] [nvarchar](50) NOT NULL,

[Роль] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Authorization] PRIMARY KEY CLUSTERED

(

[Логин] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Autentification] WITH CHECK ADD CONSTRAINT [FK\_Autentification\_Сотрудники] FOREIGN KEY([Логин])

REFERENCES [dbo].[Сотрудники] ([Паспорт сотрудника])

GO

ALTER TABLE [dbo].[Autentification] CHECK CONSTRAINT [FK\_Autentification\_Сотрудники]

GO

1. Таблица договоров

CREATE TABLE [Договор] (

[Номер договора] INTEGER NOT NULL,

[Стоимость] MONEY,

[Дата заказа] DATETIME,

[Кол-во услуг] INTEGER,

[Паспортные данные] INTEGER,

CONSTRAINT [PK\_Договор] PRIMARY KEY ([Номер договора])

);

1. Таблица песен

CREATE TABLE [Песни] (

[ID песни] INTEGER NOT NULL,

[Название песни] NVARCHAR(40) NOT NULL,

[Длина] DATETIME,

[ID альбома] INTEGER,

[Язык] NVARCHAR(40),

CONSTRAINT [PK\_Песни] PRIMARY KEY ([ID песни])

);

1. Таблица альбомов

CREATE TABLE [Альбомы] (

[ID альбома] INTEGER NOT NULL,

[Название альбома] NVARCHAR(40),

[Жанр] NVARCHAR(40),

[Общая длина] DATETIME,

[Кол-во песен] INTEGER,

EP BIT,

[Паспортные данные] INTEGER,

CONSTRAINT [PK\_Альбомы] PRIMARY KEY ([ID альбома])

);

1. Таблица выполнемых услуг

CREATE TABLE [Выполнение услуги] (

[Номер услуги] INTEGER NOT NULL,

[Срок выполнения (в раб. днях)]INTEGER,

[ID песни] INTEGER NOT NULL,

[Номер договора] INTEGER NOT NULL,

[Код услуги] INTEGER NOT NULL,

[Стоимость] MONEY,

CONSTRAINT [PK\_Выполнение услуги] PRIMARY KEY ([Номер услуги])

);

1. Таблица сотрудников

CREATE TABLE [Сотрудник] (

[Номер паспорта] INTEGER NOT NULL,

[ФИО] NVARCHAR(40),

[Должность] NVARCHAR(40),

[Дата рождения] INTEGER,

[Телефон] NVARCHAR(10),

[Пол] CHARACTER(40),

[Зарплата] MONEY,

[Номер труд. книги] INTEGER,

CONSTRAINT [PK\_Сотрудник] PRIMARY KEY ([Номер паспорта])

);

1. Таблица исполнителей

CREATE TABLE [Исполнитель] (

[Паспортные данные] INTEGER NOT NULL,

[Кол-во альбомов] INTEGER,

[Карьера] DATETIME,

[Кол-во наград] INTEGER,

[Страна] NVARCHAR(40),

[ФИО] NVARCHAR(40),

[e-mail] NVARCHAR(40),

[Телефон] NVARCHAR(10),

CONSTRAINT [PK\_Исполнитель] PRIMARY KEY ([Паспортные данные])

);

1. Таблица услуги «запись»

CREATE TABLE [Запись] (

[Перезаписи] INTEGER,

[Фрагменты] INTEGER,

[Что записывается] NVARCHAR(40),

[Код услуги] INTEGER NOT NULL,

[Сопр. инструменты] INTEGER,

[Битрейт] INTEGER,

CONSTRAINT [PK\_Запись] PRIMARY KEY ([Код услуги])

);

1. Таблица услуги «аранжировка»

CREATE TABLE [Аранжировка] (

[Название референса] NVARCHAR(40),

[Стиль/Жанр] NVARCHAR(40),

[Кол-во музыкантов] INTEGER,

[Код услуги] INTEGER NOT NULL,

CONSTRAINT [PK\_Аранжировка] PRIMARY KEY ([Код услуги])

);

1. Таблица услуги «сведение»

CREATE TABLE [Сведение] (

[Кол-во дорожек] INTEGER,

[Работа в днях] INTEGER,

[Плагин] NVARCHAR(40),

[Код услуги] INTEGER NOT NULL,

CONSTRAINT [PK\_Сведение] PRIMARY KEY ([Код услуги])

);

1. Таблица услуг, относящихся к конкретной песне

CREATE TABLE [Перечень услуг] (

[Код услуги] INTEGER NOT NULL,

[Оборудование] NVARCHAR(40),

[Название] NVARCHAR(40),

[Номер паспорта] INTEGER NOT NULL,

CONSTRAINT [PK\_Перечень услуг] PRIMARY KEY ([Код услуги])

);

ALTER TABLE [Выполнение услуги]

ADD CONSTRAINT [Выполнение услуги\_Песни\_FK\_Rule]

FOREIGN KEY ([ID песни])

REFERENCES [Песни];

ALTER TABLE [Выполнение услуги]

ADD CONSTRAINT [Выполнение услуги\_Договор\_FK\_Rule]

FOREIGN KEY ([Номер договора])

REFERENCES [Договор];

ALTER TABLE [Альбомы]

ADD CONSTRAINT [Альбомы\_Исполнитель\_FK\_Rule]

FOREIGN KEY ([Паспортные данные])

REFERENCES [Исполнитель];

ALTER TABLE [Песни]

ADD CONSTRAINT [Песни\_Альбомы\_FK\_Rule]

FOREIGN KEY ([ID альбома])

REFERENCES [Альбомы];

ALTER TABLE [Договор]

ADD CONSTRAINT [Договор\_Исполнитель\_FK\_Rule]

FOREIGN KEY ([Паспортные данные])

REFERENCES [Исполнитель];

ALTER TABLE [Запись]

ADD CONSTRAINT [Запись\_Перечень услуг\_FK\_Rule]

FOREIGN KEY ([Код услуги])

REFERENCES [Перечень услуг]

ON UPDATE CASCADE

ON DELETE CASCADE;

ALTER TABLE [Аранжировка]

ADD CONSTRAINT [Аранжировка\_Перечень услуг\_FK\_Rule]

FOREIGN KEY ([Код услуги])

REFERENCES [Перечень услуг]

ON UPDATE CASCADE

ON DELETE CASCADE;

ALTER TABLE [Сведение]

ADD CONSTRAINT [Сведение\_Перечень услуг\_FK\_Rule]

FOREIGN KEY ([Код услуги])

REFERENCES [Перечень услуг]

ON UPDATE CASCADE

ON DELETE CASCADE;

ALTER TABLE [Выполнение услуги]

ADD CONSTRAINT [Выполнение услуги\_Перечень услуг\_FK\_Rule]

FOREIGN KEY ([Код услуги])

REFERENCES [Перечень услуг];

ALTER TABLE [Перечень услуг]

ADD CONSTRAINT [Перечень услуг\_Сотрудник\_FK\_Rule]

FOREIGN KEY ([Номер паспорта])

REFERENCES [Сотрудник];

* + - 1. Скрипты хранимых процедур

1. Хранимая процедура Dogovor принимает на вход значение номера договора и на основе этих данных и данных, имеющихся в таблицах «Выполнение услуги» и «Договоры» заполняет в таблице «Договоры» ячейку стоимости и количества услуг для того договора, номер которого передавался в процедуру.

ALTER PROCEDURE Dogovor(@Dog\_count NVARCHAR(10))

AS

--Узнаем количество услуг, которые должны быть выполнены по данному договору

DECLARE @kolvo\_uslug INT = (SELECT COUNT([Выполнение услуги].[Номер услуги])

FROM dbo.[Выполнение услуги], dbo.Договоры

WHERE [Выполнение услуги].[Номер договора] = Договоры.[Номер договора] AND Договоры.[Номер договора] = @Dog\_count);

--Узнаем общую стоимость договора

DECLARE @Sum MONEY = (SELECT SUM([Выполнение услуги].Стоимость)

FROM dbo.[Выполнение услуги], dbo.Договоры

WHERE Договоры.[Номер договора] = @Dog\_count AND [Выполнение услуги].[Номер договора] = @Dog\_count);

--Объявляем переменную, отвечающую за заказчика

DECLARE @zak NVARCHAR(10) = (SELECT Договоры.[Паспорт заказчика] FROM dbo.Договоры WHERE Договоры.[Номер договора] = @Dog\_count);

--Объявляем переменную, отвечающую за количество договоров, которые были заключены с данным заказчиком

DECLARE @Col\_Dog INT = (SELECT COUNT(Договоры.[Номер договора] ) FROM dbo.Договоры WHERE dbo.Договоры.[Паспорт заказчика] = @zak);

--Если количество таких договоров больше двух...

IF (@Col\_Dog > 2)

BEGIN

--...Уведомляем о том, что на текущий договор предоставляется 10-процентная скидка

--и обновляем записи с количеством услуг и стоимостью договоро в таблице Договоры

PRINT 'Вам была сделана скидка на 3й заказ'

UPDATE dbo.Договоры SET Договоры.[Кол-во услуг] = @kolvo\_uslug, Договоры.Стоимость = @Sum\*0.9 WHERE Договоры.[Номер договора] = @Dog\_count;

END

ELSE

--Иначе просто обновляем записи с количеством услуг и стоимостью договоро в таблице Договоры

UPDATE dbo.Договоры SET Договоры.[Кол-во услуг] = @kolvo\_uslug, Договоры.Стоимость = @Sum WHERE Договоры.[Номер договора] = @Dog\_count;

GO

1. Хранимая процедура Nalog принимает на вход значение процента налога, взимаемого с дохода физических лиц, работающих на студии. Столбец Зарплата в таблице «Сотрудники» изменияется. Зарплата каждого сотрудника уменьшается на соответствующий процент

CREATE PROCEDURE Nalog(@PerCent INT)

AS

UPDATE dbo.Сотрудники

SET Сотрудники.Зарплата = Сотрудники.Зарплата - Сотрудники.Зарплата\*CONVERT(MONEY,@PerCent)/100;

GO

* + - 1. Скрипты определяемых пользователем функций

1. Определяемая пользователем функция, возвращающая табличное значение, PeriodSell принимает на вход два параметра – начальная и конечная даты, обозначающие период времени, за который пользователь желает узнать общую выручку студии звукозаписи. Для этого внутри функции используются данные о дате заказа из таблицы «Договоры». Стоимость всех договоры, которые были заключены в обозначенный входными параметрами период, суммируется и возвращается.

CREATE FUNCTION PeriodSell(@Date1 DATE, @Date2 DATE)

RETURNS TABLE

AS RETURN

(SELECT SUM(Договоры.Стоимость) AS "Salary" FROM dbo.Договоры WHERE Договоры.[Дата заключения] BETWEEN @Date1 AND @Date2);

GO

# 3 Алгоритмическое обеспечение системы

Данный алгоритм позволяет избежать наличия ошибок при заполнении таблицы Договор, путем «автоматического» заполнения данных о стоимости конкретного договора и количестве услуг, которые предоставляет студия согласно условий данного договора. На вход алгоритм принимает номер интересующего договора. Так же данным алгоритмом реализуется своеобразная программа лояльности для клиентов. Производится проверка количества договоров для заказчика, указанного в данном договоре. В случае, если договор является третьим или больше, то его конечная стоимость уменьшается на 10%, о чем так же уведомляется пользователь.

## 3.1 Общая характеристика

Заполнение полей стоимости и количества услуг в таблице Договоры для каждого определенного договора, номер которого указывается в принимаемых параметрах процедуры. Так же идет скидка на услуги с заключения третьего договора.

## 3.2. Используемые данные

Атрибуты таблицы Выполнение услуги: Номер услуги, Стоимость, Номер Договора. Атрибуты таблицы Договор: Номер Договора, Паспорт Заказчика.

## 3.3. Результаты выполнения

Обновление таблицы Договоры, изменение значений атрибутов этой таблицы – Стоимость и Количество услуг.

### 3.4 Логическое описание

## КурсовойПроцедура

# 4 Прикладное программное обеспечение системы

## 4.1 Общая характеристика прикладного программного обеспечения

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

В системе присутствует аутентификация, реализованная с помощью дополнительной таблицы в базе данных проекта.

Исключительные ситуации отслеживаются и обрабатываются с помощью блоков try catch и специальных представлений, указывающих на возникновение ошибки.

Базовый функционал всех таблиц:

1. Просмотр записей
2. Редактирование записей
3. Удаление записей
4. Добавление записей

Реализован поиск в таблице «Песни». Можно найти песню по названию, введенному в специальное средство управления TextBox. По нажатию кнопки «Искать!» с помощью точечной нотации находится все песни с указанным названием.

Реализована фильтрация в талице «Альбомы». Фильтрация производится по жанрам. С помощью элемента управления DropDownList можно выбрать один из четырех жанров – «Рок», «Инди», «Поп» и «Классика». По нажатию кнопки «Фильтровать!» пользователю будет предоставлена информация об альбомах с выбранным жанром.

Реализована фильтрация в таблице «Сотрудники». Фильтрация происходит по должности. На выбор представлены три должности, указанные в элементе управления DropDownList – «Звукорежиссер», «Менеджер», «Инженер». По нажатию кнопки «Фильтровать!» пользователю будет предоставлена информация обо всех сотрудниках с выбранной должностью.

В базе данных присутствуют:

1. Хранимые процедуры – 2.
   * Процедура Dogovor позволяет избежать ошибок в установки конечной стоимости договора и количества услуг, согласно указанному договору путем самостоятельного подсчета стоимости и количества услуг, исходя из данных таблицы «Выполнение услуги». Так же она реализует скромную программу лояльности для постоянных клиентов, предоставляя скидку в 10% для третьего и последующих заказов.
   * Процедура Naloog обновляет в таблице «Сотрудники» зарплату каждого сотрудника студии с учетом налоговых вычет, переданных в качестве параметра в процедуру.
2. Определяемые пользоватеоем функции (табличные) – 1.
   * Функция PeriodSell возвращает суммарную стоимость договоров, заключенных между звукозаписывающей студией и клиентами, за указанный период, который передается в функцию параметрами – начало периода и конец периода.

Для таких атрибутов, как «Телефон», «Эл. Почта», «Номер трудовой книги» использовались индексы для поддержки уникальности этих полей. Уникальность требуется по логическим соображениям.

Количество таблиц – 10;

Количество атрибутов – 42;

Количество типов связей – 3.

## 4.2 Структура и состав прикладного программного обеспечения

*Properties* – в этой папке содержится единственный файл, который отвечает за сведения о сборке проекта.

*Ссылки* – ссылки, используемые для работы проекта.

*App Data* – в этой папке хранятся файлы базы данных, которые используются приложением.

*App Start* – в этой папке хранится ряд файлов, отвечающих за настроки конфигурации для проекта.

*Content* – содержит все файлы, которые не относятся к C# и JavaScript. Сюда относятся файлы css и картинки, которые отображаются на страницах проекта.

*Controllers* – в этой папке содержатся модули всех контроллеров проекта.

*Fonts* – в этой папке хранятся файлы дополнительных нестандартных шрифтов, которые могут использоваться приложением.

*Models* – в этой папке хранятся файлы моделей для приложения.

*Scripts* – в этой папке содержатся файлы скриптов JavaScript.

*Views* – в этой папке хранятся файлы представлений, сгруппированных по папкам с именем контроллера, с которым эти представления связаны.

*Favicon.ico* – иконка, отображаемая при работе проекта во вкладке браузера.

*Global.asax* – данный файл запускается при старте работы приложения. Выполняет начальную инициализацию проекта.

*Packages.Config* – данный файл содержит в себе пакеты NuGet, установленные в проекте.

*Web.Config* – данный файл отвечает за конфигурацию проекта.

В папке *Controllers* содержатся следующие файлы контроллеров:

1. *AuthController*

Контроллер отвечает за корректную атентификацию пользователя, определение его как Администратора или Пользователя с последующим направлением на соответствующие «линейки» представлений.

Значимых строк кода - 4

1. *HomeController*

Контроллер отвечает за вывод главной страницы на экран для аутентифицированного и неаутентифицированного пользователя.

Значимых строк кода - 2

1. *АльбомыController*

Контроллер отвечает за вывод на экран информации об альбомах, записанных студией звукозаписи. Для пользователя, аутентифицированного как Администратор, имеется возможность изменения записи об альбомах, удаления записи и добавления новой записи. Описанные выше возможности далее доступны для Администратора во всех таблицах. Пользователь и Гость могут просматривать лишь частичные записи в этой таблице, причем Пользователь может выполнить фильтрацию по жанру и детально просмотреть каждую запись.

Значимых строк кода - 41

1. *АранжировкаController*

Контроллер отвечает за отображение данных, связанных с предоставляемой услугой аранжировки песен. Данная информация недоступна ни для кого, кроме Администратора.

Значимых строк кода - 32

1. *Выполнение\_услугиController*

Контроллер отвечает за отображение информации обо всех выполненных студией услугах. Там же отображается номер договора, к которому привязана эта услуга, песня, для которой выполнялась услуга, и ее итоговая стоимость. Данная информация недоступна ни для кого, кроме Администратора.

Значимых строк кода - 40

6) *ДоговорыController*

Контроллер отвечает за отображение подробной информации о договорах, заключенных между исполнителями и студией звукозаписи. Данная информация недоступна ни для кого, кроме Администратора.

Значимых строк кода - 42

1. *ЗаписьController*

Контроллер отвечает за отображение данных, связанных с предоставляемой услугой записи фрагментов песен. Данная информация недоступна ни для кого, кроме Администратора.

Значимых строк кода - 31

1. *ИсполнительController*

Контроллер отвечает за отображение информации об исполнителях, которые работали со студией звукозаписи. Они же и являются клиентами студии. Данная информация доступна для Пользователя и Гостя частично, причем Пользователь имеет возможность детально просмотреть каждую запись.

Значимых строк кода - 28

1. *Перечень\_услугController*

Контроллер отвечает за отображение информации об услугах студии звукозаписи, которые связаны с конкретным сотрудником организации. Эти данные доступны лишь для Администратора.

Значимых строк кода - 46

1. *ПесниController*

Контроллер отвечает за отображение информации о песнях, записанных студией звукозаписи. Частично эту информацию может просматривать Пользователь и Гость могут просматривать записи лишь частично. Пользователь может просматривать записи детально, также ему доступен поиск по названию песни.

Значимых строк кода - 37

1. *СведениеController*

Контроллер отвечает за отображение данных, связанных с предоставляемой услугой сведения фрагментов песен. Данная информация недоступна ни для кого, кроме Администратора.

Значимых строк кода - 33

1. *СотрудникиController*

Контроллер отвечает за отображение информации о сотрудниках, работающих в звукозаписывающей студии. Пользователь, не являющийся Администратором, может частично просматривать эту информацию. Так же в контроллере реализована фильтрация сотрудников по должности.

Значимых строк кода - 43

## 4.3 Особенности реализации и сопровождения

При реализации проекта была использована архитектура ASP.NET MVC. Создание типовых контроллеров значительно упростило работу над проектом, а технология Entity Framework позволила избежать муторного создания представлений для редактирования, удаления, создания и отображения элементов модели, что позволило создать удобный пользовательский интерфейс без лишних функций. Такой подход, во-первых, облегчил нагрузку на сервер, а, во-вторых, сделал интерфейс приложения простым и понятным пользователю.

Аутентификация реализована с помощью дополнительной таблицы в базе данных SQL Server. В данной таблице указываются паспортные данные каждого сотрудника студии звукозаписи в качестве логина, пароль, для входа сотрудника в систему, и роль сотрудника – Администратор или Пользователь. После того, как пользователь, который хочет войти в систему, вводит данные в специальные элементы управления на странице, происходит проверка на наличие в таблице аутентификации введенных логина и пароля. Если такой пользователь найден, то проверяется его роль, в соответствии с которой пользователь далее перенаправляется на определенные представления для Администратора или для аутентифицированного Пользователя. Если же такая пара значений «логин»-«пароль» не была найдена в соответствующей таблице, пользователь информируется о неправильно введенных данных, а также ему предлагается ввести данные снова.

В ходе работы над проектом была создана EDM модель, которая использовалась для последующего построения программных модулей. Это позволило обеспечить их корректную работу.

# 5 Руководство пользователя

## 5.1 Общие сведения

Система представляет собой обобщенную модель студии звукозаписи. Модель создана для облегчения работы с большими объемами структурированных данных. В проекте был реализован предельно простой интерфейс для удобста работы с данными.

Система позволяет хранить данные о заказах в студии звукозаписи, клиентах студии и основных выполняемых услугах. Так же в функционал системы входит возможность добавления, изменения и удаления записей. Присутствует функция для исправления неточностей при подстчете итоговой стоимости договора, фильтрация сотрудников по должности и поска песни по названию.

## 5.2 Порядок и особенности работы

1. Запускаем проект. Первое, что видит пользователь – стартовая страница с возможностью аутентифицироваться или продолжить работу в системе в качестве «Гостя». (рис. 5.1).

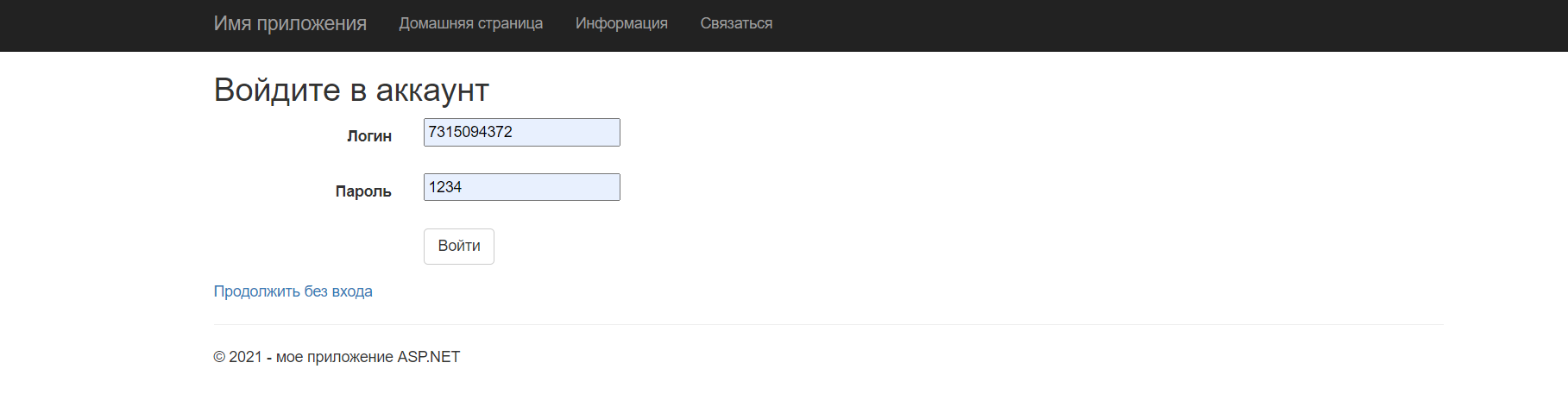


Рисунок 5.1 – Главная страница

2. У Администратора есть своя «стартовая страница» со ссылками на все таблицы базы данных (рис. 5.2). С этой страницы он может перейти к любой таблице для работы с ней.

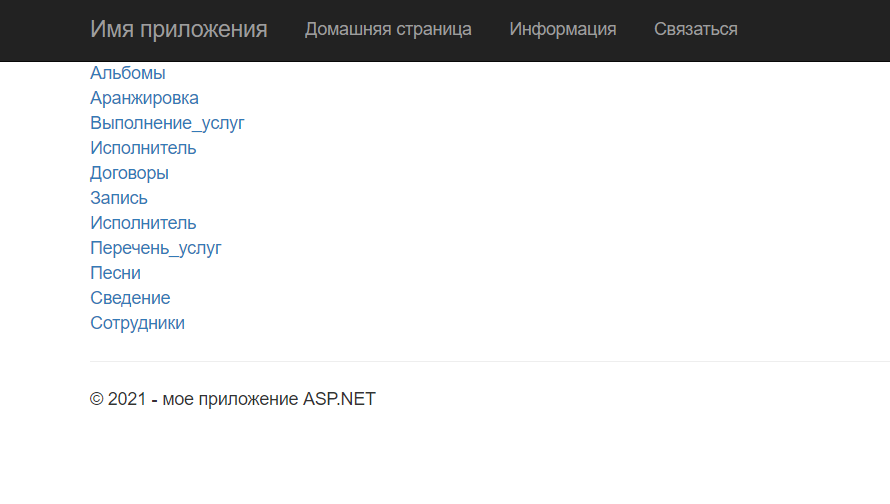


Рисунок 5.2 – Ссылки на таблицы

3. Таблицы для Администратора выглядят следубщим образом (рис 5.3) Рассмотрим функционал, доступный Администратору на примере таблицы «Выполнение услуги». Администратор может добавлять записи в таблицу (рис. 5.4), просматривать детально конкретную запись (рис. 5.5), удалять какую-либо запись (рис. 5.6)

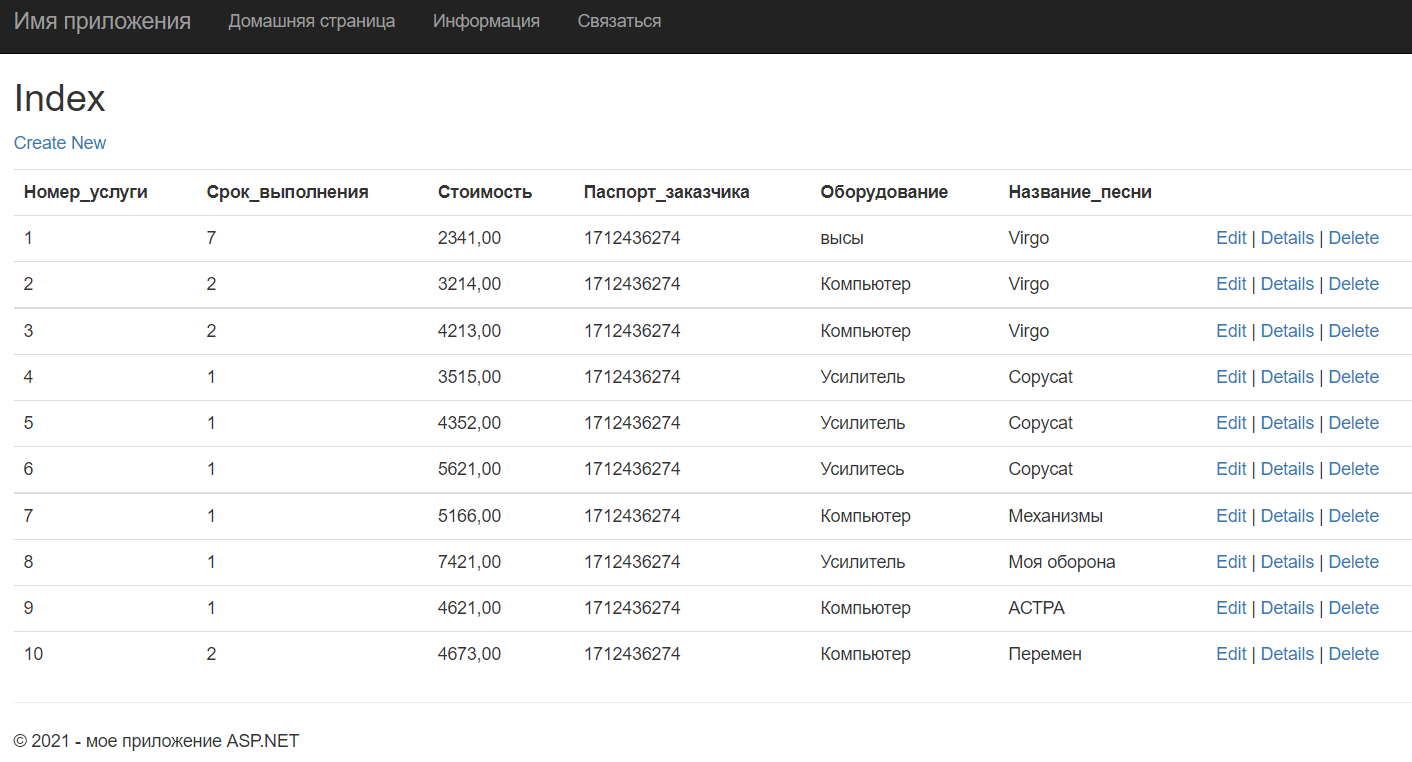


Рисунок 5.3 – Просмотр записей в табличном виде

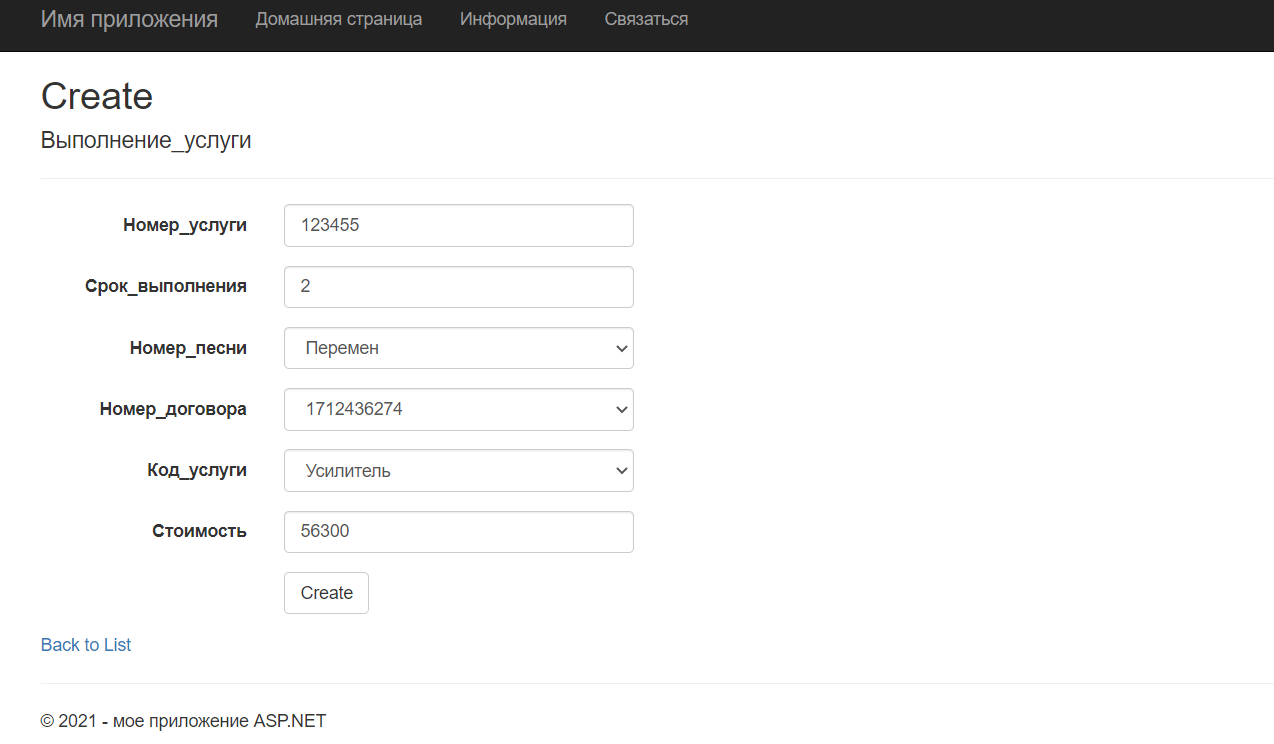


Рисунок 5.4 – Добавление новой записи

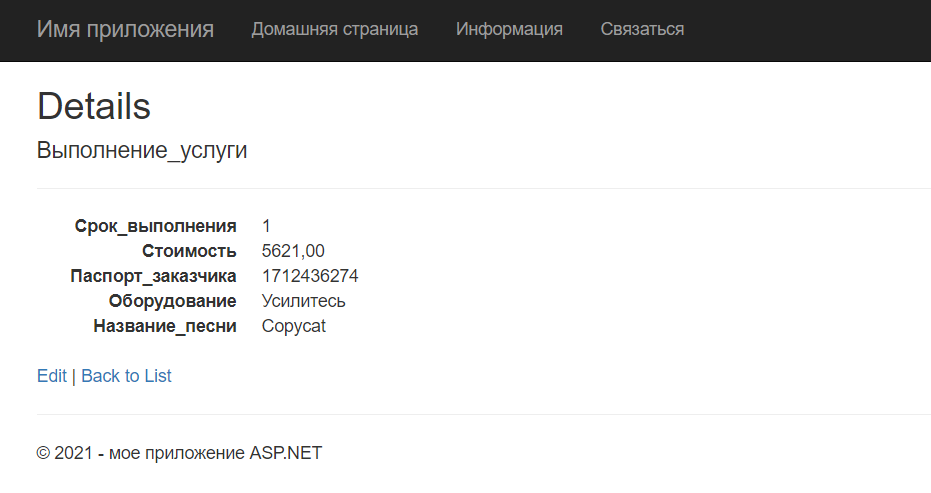


Рисунок 5.5 – Просмотр данных о записи

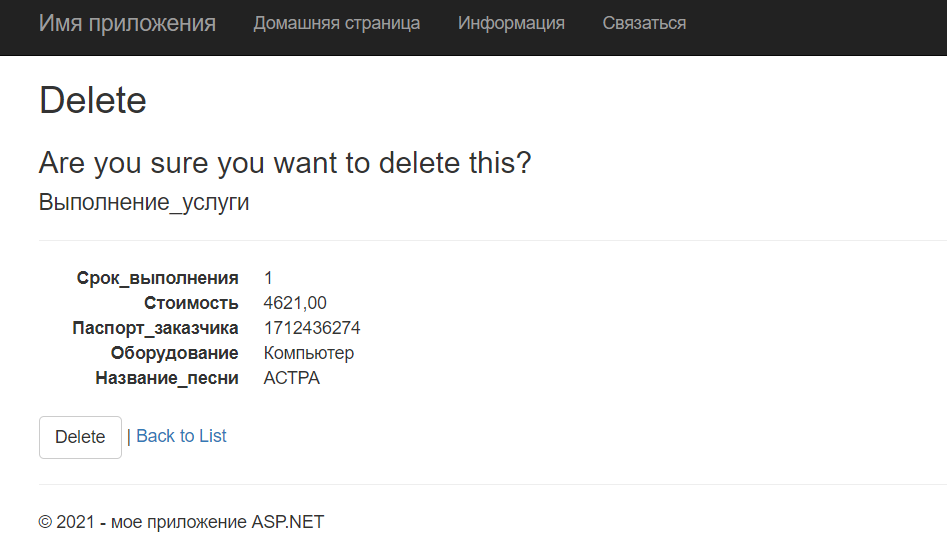


Рисунок 5.6 – Удаление записи

4. Весь перечисленный выше функционал доступен Администратору для всех таблиц базы данных. Так же Администратор может выполнять фильтрацию Сотрудников по должности (рис. 5.7) и фильтрацию Альбомов по жанру (рис. 5.9). Для этого необходимо выбрать желаемый критерий из выпадающего списка наверху страницы и нажать кнопку «Фильтровать!». Пустая строка вернет весь изначальный список.

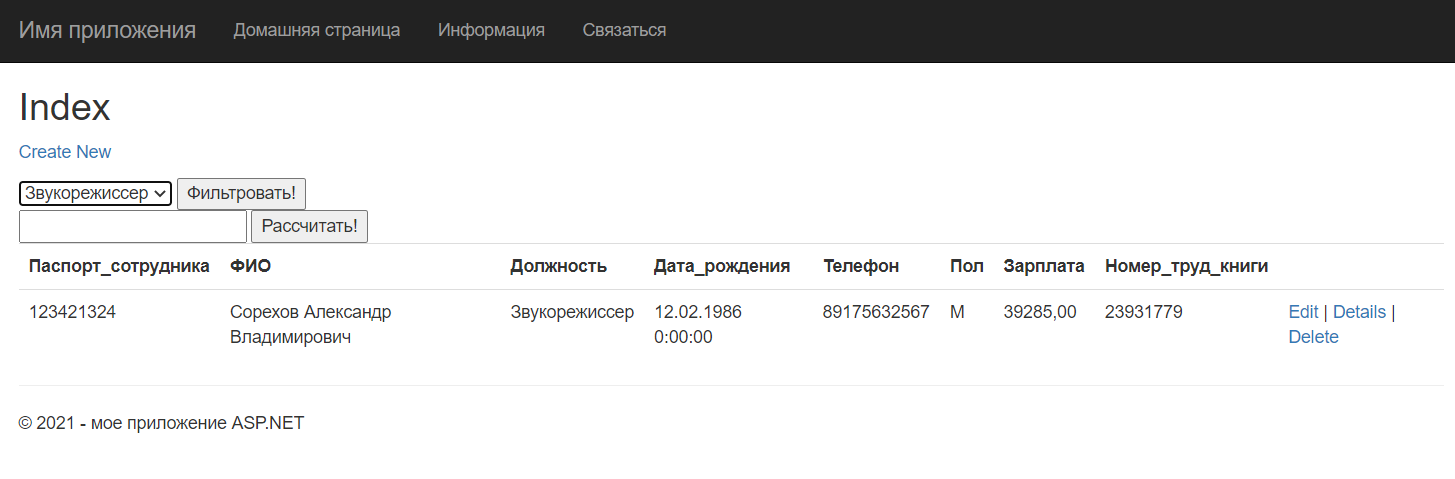


Рисунок 5.8 – Фильтрация Сотрудников по должности

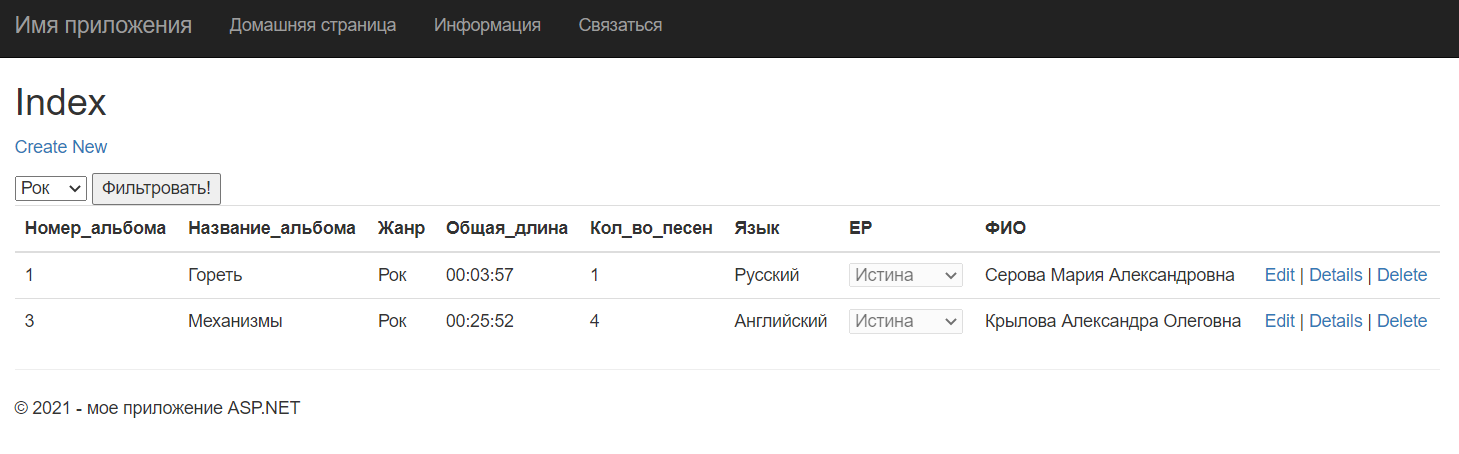


Рисунок 5.9 – Фильтрация Альбомов по жанру.

Также для Администратора доступна возможность искать песню по ее названию в соответствующей таблице (рис. 5.10). Для этого небходимо ввести название интересующей песни и нажать кнопку «Искать!».

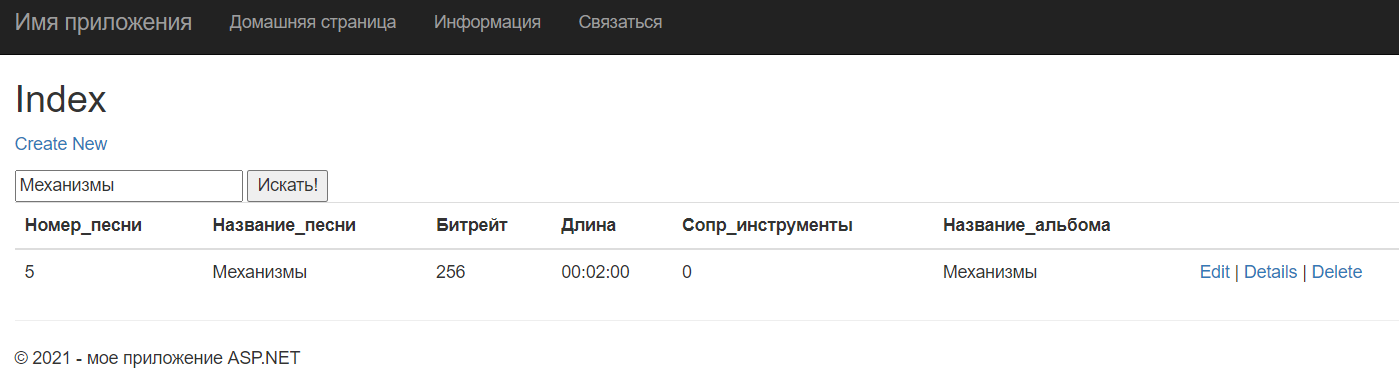


Рисунок 5.10 – Поиск песни по названию.

Помимо этого Администратор может изменить зарплату Сотрудника, если изменяется налог (рис. 5.11). Для этого нужно внести процент налогового вычета в специальное поле вверху страницы и нажать кнопку «Рассччитать!»

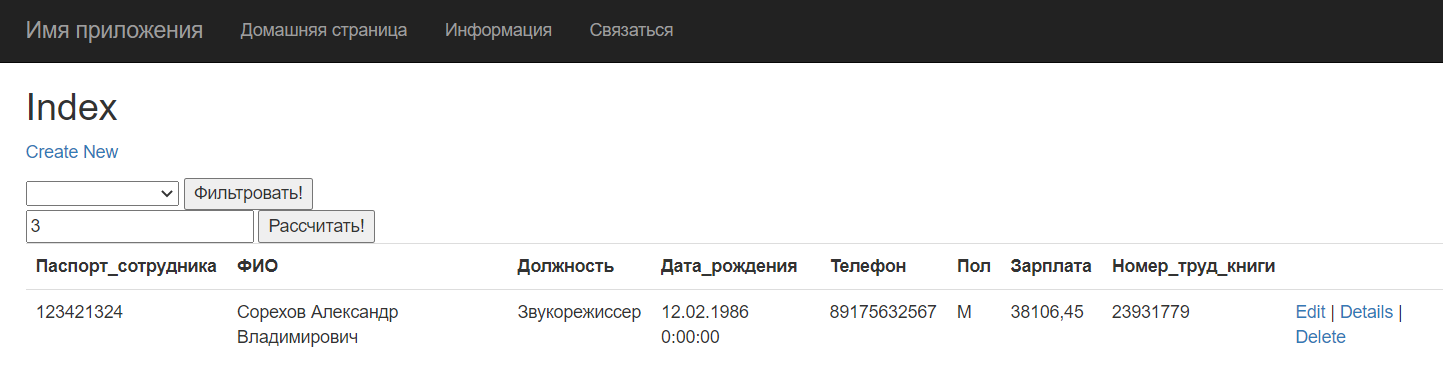


Рисунок 5.11 – Изменение зарплаты Сотрудника

Администратор может проверить введенные данные в таблице Договоры путем введения определенного номера договора в специальное поле в верхней части страницы. В этом случае стоимость договора будет пересчитана в сответствие со стоимостью всех выполняемых услуг. Так же будет изменено количество услуг на реальное значение из таблицы Услуг. (рис. 5.12)

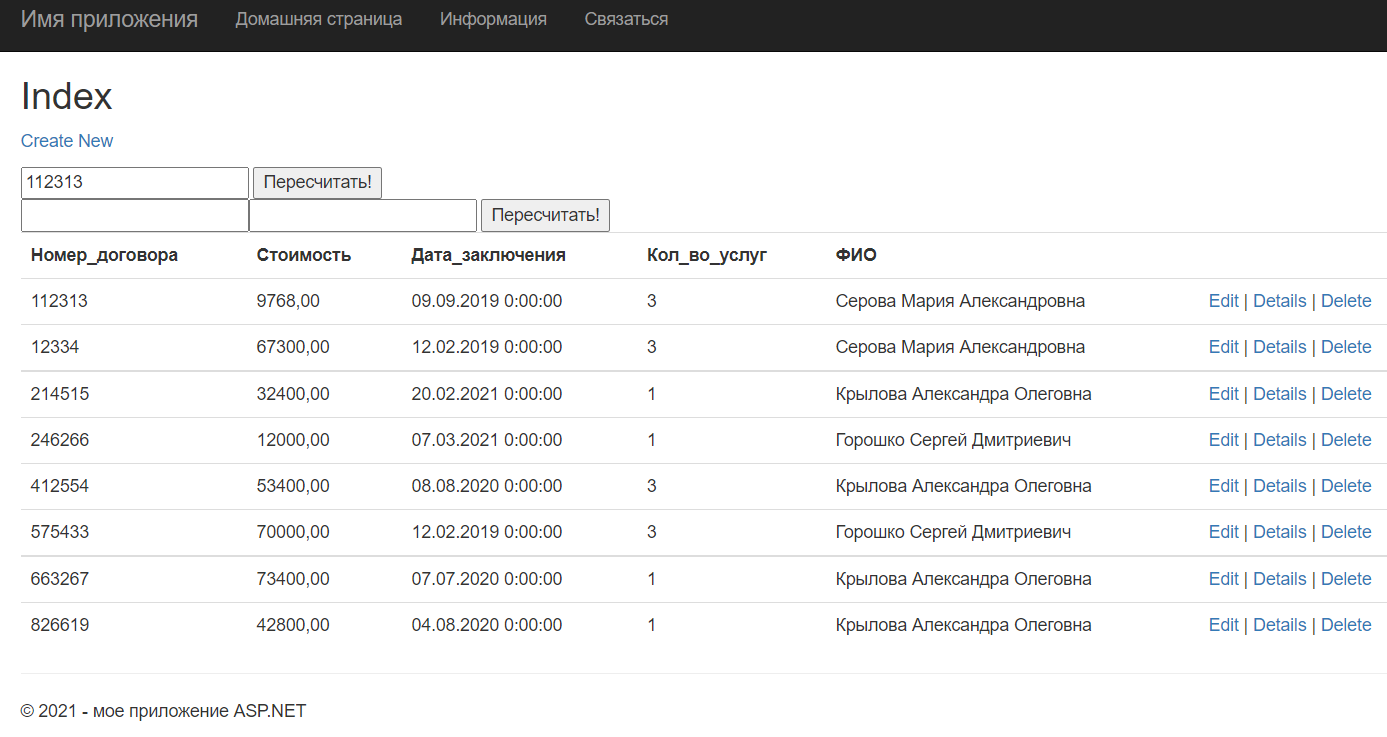


Рисунок 5.12 – Коррекция данных о Договоре

Администратор может узнать всю выручку с договоров за опредленный период (рис. 5.13), рамки которого нужно ввести в два специальных поля в верхней части страницы.

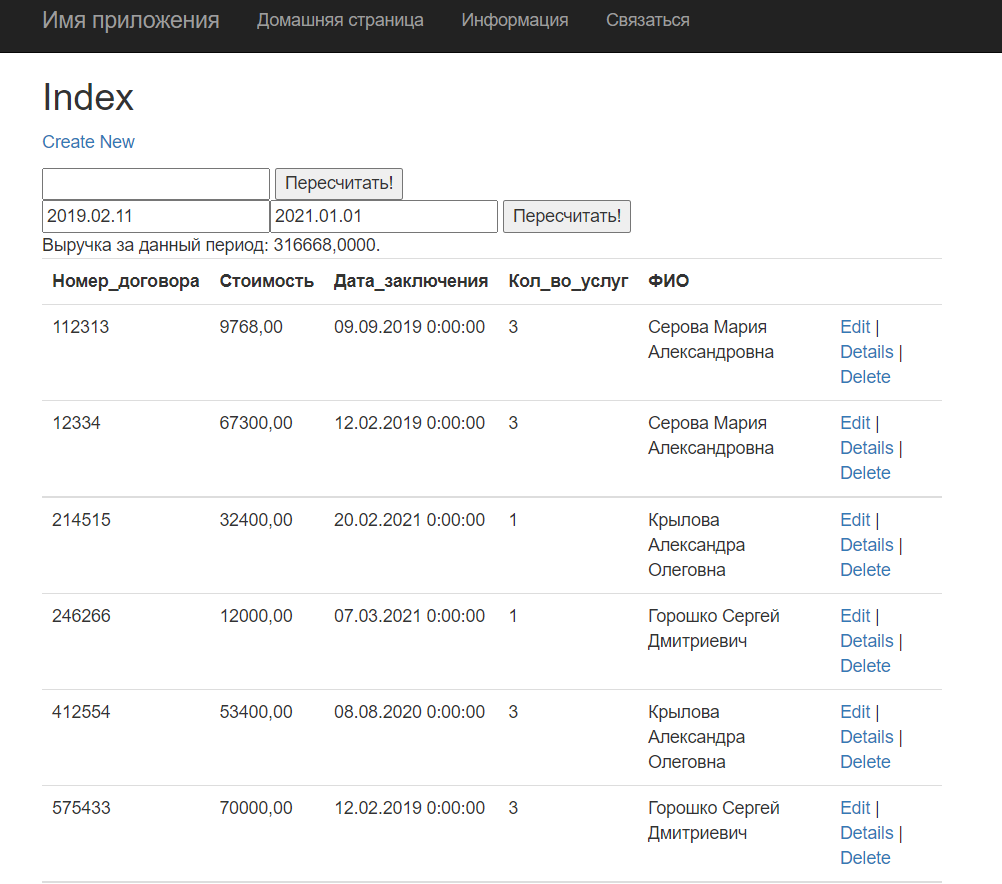


Рисунок 5.13 – Вычисление суммарной выручки за произвольныый период.

5. Для Аутентифицированного Пользователя доступно меньшее количество таблиц (рис. 5.14). В них он не может добавлять, изменять или редактировать записи, однако он также может просматривать их детально (рис. 5.15)

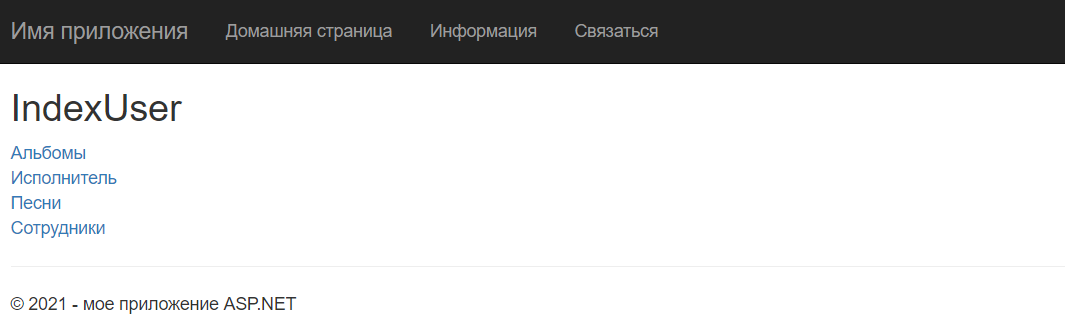


Рисунок 5.14 – Таблицы, доступные Аутентифицированному Пользователю.

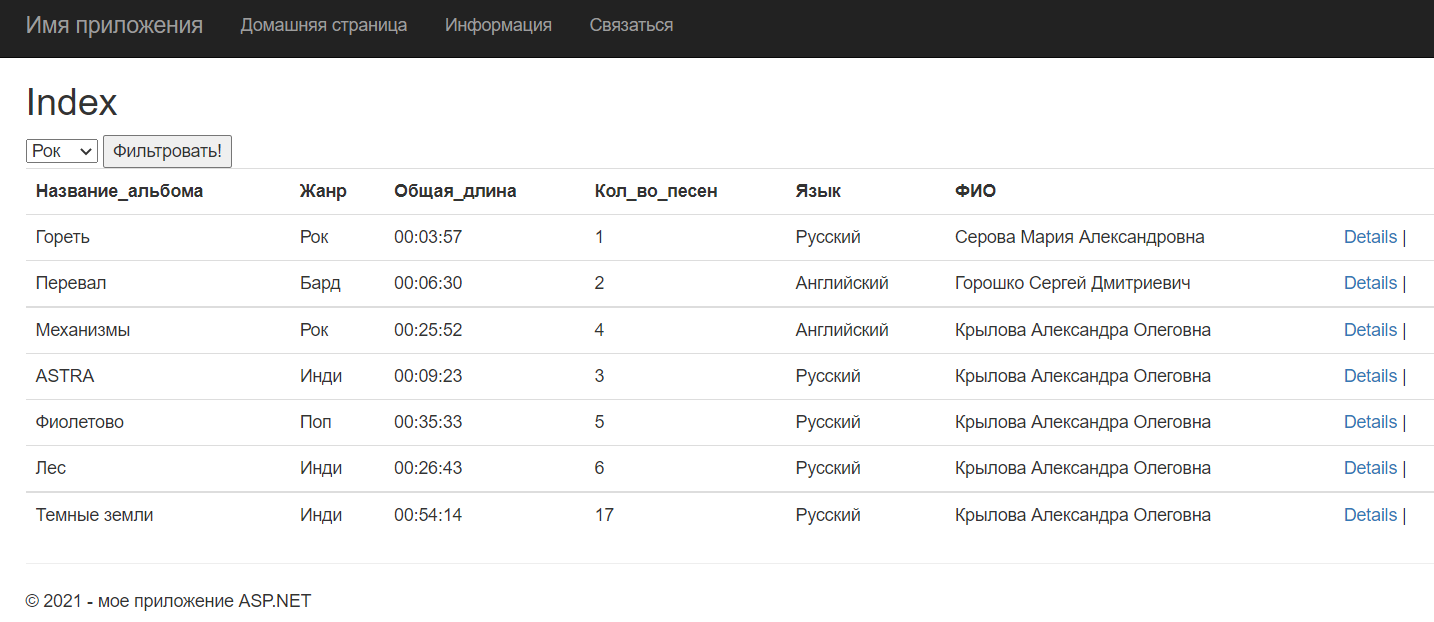


Рисунок 5.15 – Таблица для Аутентифицированного Пользователя

Аутентифицированному пользователю также, как и Администратору, доступны фильтрация и поиск в соответствующих таблицах.

6. Неаутентифицированному Пользователю (Гостю) доспупно еще меньшее количество таблиц для просмотра (рис. 5.16). Так же он не может использовать фильтрацию и поиск, в отличие от Аутентифицированного Пользователя. Еще одно ограничение для Гостя - отсутствие возможности просмотреть детально подробности и записи в таблице (рис. 5.17)

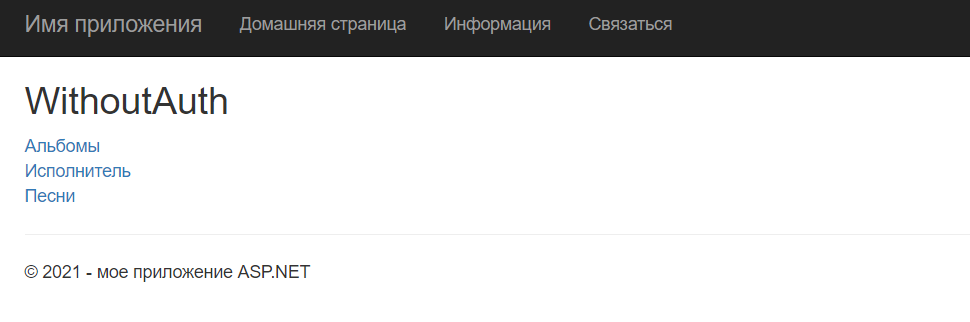


Рисунок 5.16 – Ссылки на таблицы для Гостя

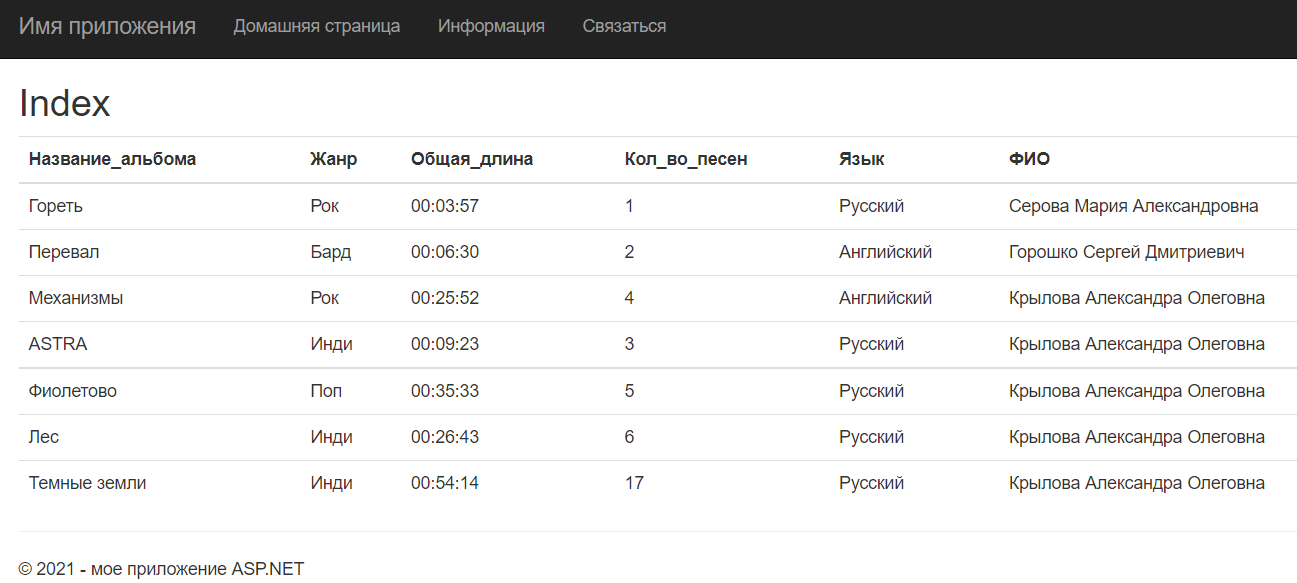


Рисунок 5.17 – Вид таблицы для Гостя

## 5.3 Исключительные ситуации

1) Исключительная ситуация возникает в случае, если при аутентификации пользователь вводит неверные данные, и приложение не может обаруждить соответствующие данные в таблице аутентификации в SQL Server. В этом случае пользователь будет уведомлен о том, что были введены неверные данные для входа. Будет преедложен повторный вход в систему.

2) Исключительная ситуация возникает при попытке добавить сотрудника или исполнителя с уже имеющимся телефоном или паспортными данными. Система уведомит Администратора об ошибке, а такая запись не будет сохранена в базе данных.

3) Исключительная ситуация возникает при попытке добавить сотрудника с трудовой книгой, номер которой совпадает с номером трудовой книги иного сотрудника, уже имеющегося в базе. В таком случае, как и в предыдущем, дминистратору будет показано уведомление об ошибке, а запись не сохранится в базе данных.

4) Исключительная ситуация возникает при попытке удалить Исполнителя, у которого имеются записи об Альбомах или Договорах. В этом случае Пользователь будет информирован о появлении исключения и перенаправлен на представление с информацией об ошибке.

5) Исключительная ситуация возникает при попытке добавить запись об одной из услуг Аранжировка, Сведение или Запись, если, при этом, параметр Код Услуги, который указывается в одной из этих трех таблиц, не существует в таблице Перечень Услуг. В этом случае, как и ранее, Пользователь будет информирован о появлении исключения и перенаправлен на представление с информацией об ошибке.

# Список использованных источников

1. Работники студии звукозаписи. Звукорежиссер// Sintez-audio.ru : сайт. – URL: <http://sintez-audio.ru/news/117-2015-10-23-07-26-27> (дата обращения: 04.05.2021). – Текст : электронный
2. Какое оборудование нужно для студии звукозаписи? (собираем базу) // Arefyevstudio.com : сайт. – URL: <https://arefyevstudio.com/2017/04/28/kakoe-oborudovanie-nuzhno-dlya-studii-zvukozapisi/> (дата обращения: 04.05.2021). – Текст : электронный
3. Работники студии звукозаписи. Аранжировщик// Sintez-audio.ru : сайт. – URL: <http://sintez-audio.ru/news/125-2015-11-23-06-44-25> (дата обращения: 04.05.2021). – Текст : электронный
4. Инженерно-технический персонал студии звукозаписи// Sintez-audio.ru : сайт. – URL: <http://sintez-audio.ru/news/126-2015-12-09-07-12-31> (дата обращения: 14.06.2021). – Текст : электронный
5. Фриман, А./ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов. - 2013. – URL : <https://smarly.net/pro-asp-net-mvc-4> (дата обращения 14.06.2021). – Текст :электронный
6. Руководство по ASP.NET MVC 5// metanit.com : сайт. – 2017. – URL : <https://metanit.com/sharp/mvc5/> (дата обращения 14.06.2021). – Текст :электронный
7. Родионов В. В. Программирование хранимых процедур, определяемых пользователем функций и триггеров : методические указания. – 2021. (дата обрвщения 01.05.2021). – Текст : непосредственный
8. Родионов В. В. Язык интегрированных запросов LINQ : методические указания. – 2021. (дата обрвщения 01.05.2021). – Текст : непосредственный

# Приложение А. Исходные тексты программных модулей

**AuthController.cs**

using KourseWork.Models;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Security;

namespace KourseWork.Controllers

{

public class AuthController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

public ActionResult Login()

{

return View();

}

public ActionResult IndexUser()

{

return View();

}

public ActionResult IndexAdmin()

{

return View();

}

public ActionResult WithoutAuth()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Login(string Pasport, string Password)

{

var User = db.Autentification.FirstOrDefault(p => p.Логин == Pasport && p.Пароль == Password);

if (User != null)

{

if (User.Роль == "admin")

{

return RedirectToAction("Index", "Home");

}

else if (User.Роль == "user")

{

return RedirectToAction("IndexUser", "Auth");

}

else

{

ModelState.AddModelError("", "Пользователя с таким логином и паролем нет");

}

}

return View();

}

}

}

**HomeController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace KourseWork.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

}

}

**АльбомыController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class АльбомыController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Альбомы

public ActionResult Index()

{

var альбомы = db.Альбомы.Include(а => а.Исполнитель);

return View(альбомы.ToList());

}

[HttpPost]

public ActionResult Index(string janr)

{

IEnumerable<Альбомы> альбомы = db.Альбомы.AsEnumerable();

if (!string.IsNullOrEmpty(janr))

{

альбомы = db.Альбомы.Where(x => x.Жанр == janr);

}

return View(альбомы.ToList());

}

public ActionResult UserIndex(string janr)

{

IEnumerable<Альбомы> альбомы = db.Альбомы.AsEnumerable();

if (!string.IsNullOrEmpty(janr))

{

альбомы = db.Альбомы.Where(x => x.Жанр == janr);

}

return View(альбомы.ToList());

}

public ActionResult GuestIndex()

{

IEnumerable<Альбомы> альбомы = db.Альбомы.AsEnumerable();

return View(альбомы.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Альбомы/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return RedirectToAction("Error400", "Error");

}

Альбомы альбомы = db.Альбомы.Find(id);

if (альбомы == null)

{

return RedirectToAction("Error404", "Error");

}

return View(альбомы);

}

// GET: Альбомы/Create

public ActionResult Create()

{

ViewBag.Паспортные\_данные\_исполнителя = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО");

return View();

}

// POST: Альбомы/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\_альбома,Название\_альбома,Жанр,Общая\_длина,Кол\_во\_песен,Язык,EP,Паспортные\_данные\_исполнителя")] Альбомы альбомы)

{

if (ModelState.IsValid)

{

try

{

db.Альбомы.Add(альбомы);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

ViewBag.Паспортные\_данные\_исполнителя = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", альбомы.Паспортные\_данные\_исполнителя);

return View(альбомы);

}

// GET: Альбомы/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Альбомы альбомы = db.Альбомы.Find(id);

if (альбомы == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Паспортные\_данные\_исполнителя = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", альбомы.Паспортные\_данные\_исполнителя);

return View(альбомы);

}

// POST: Альбомы/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\_альбома,Название\_альбома,Жанр,Общая\_длина,Кол\_во\_песен,Язык,EP,Паспортные\_данные\_исполнителя")] Альбомы альбомы)

{

if (ModelState.IsValid)

{

try

{

db.Entry(альбомы).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Паспортные\_данные\_исполнителя = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", альбомы.Паспортные\_данные\_исполнителя);

return View(альбомы);

}

// GET: Альбомы/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Альбомы альбомы = db.Альбомы.Find(id);

if (альбомы == null)

{

return RedirectToAction("Error404", "Error");

}

return View(альбомы);

}

// POST: Альбомы/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

try

{

Альбомы альбомы = db.Альбомы.Find(id);

db.Альбомы.Remove(альбомы);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**АранжировкаController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class АранжировкаController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Аранжировка

public ActionResult Index()

{

var аранжировка = db.Аранжировка.Include(а => а.Перечень\_услуг);

return View(аранжировка.ToList());

}

// GET: Аранжировка/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Аранжировка аранжировка = db.Аранжировка.Find(id);

if (аранжировка == null)

{

return RedirectToAction("Error404", "Error");

}

return View(аранжировка);

}

// GET: Аранжировка/Create

public ActionResult Create()

{

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование");

return View();

}

// POST: Аранжировка/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 = "Код\_услуги,Референс,Жанр,Кол\_во\_музыкантов")] Аранжировка аранжировка)

{

if (ModelState.IsValid)

{

try

{

db.Аранжировка.Add(аранжировка);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка создания записи!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", аранжировка.Код\_услуги);

return View(аранжировка);

}

// GET: Аранжировка/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Аранжировка аранжировка = db.Аранжировка.Find(id);

if (аранжировка == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", аранжировка.Код\_услуги);

return View(аранжировка);

}

// POST: Аранжировка/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 = "Код\_услуги,Референс,Жанр,Кол\_во\_музыкантов")] Аранжировка аранжировка)

{

if (ModelState.IsValid)

{

try

{

db.Entry(аранжировка).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", аранжировка.Код\_услуги);

return View(аранжировка);

}

// GET: Аранжировка/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Аранжировка аранжировка = db.Аранжировка.Find(id);

if (аранжировка == null)

{

return RedirectToAction("Error404", "Error");

}

return View(аранжировка);

}

// POST: Аранжировка/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try

{

Аранжировка аранжировка = db.Аранжировка.Find(id);

db.Аранжировка.Remove(аранжировка);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**Выполнение\_услугиController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class Выполнение\_услугиController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Выполнение\_услуги

public ActionResult Index()

{

var выполнение\_услуги = db.Выполнение\_услуги.Include(в => в.Договоры).Include(в => в.Перечень\_услуг).Include(в => в.Песни);

return View(выполнение\_услуги.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Выполнение\_услуги/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Выполнение\_услуги выполнение\_услуги = db.Выполнение\_услуги.Find(id);

if (выполнение\_услуги == null)

{

return RedirectToAction("Error404", "Error");

}

return View(выполнение\_услуги);

}

// GET: Выполнение\_услуги/Create

public ActionResult Create()

{

ViewBag.Номер\_договора = new SelectList(db.Договоры, "Номер\_договора", "Паспорт\_заказчика");

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование");

ViewBag.Номер\_песни = new SelectList(db.Песни, "Номер\_песни", "Название\_песни");

return View();

}

// POST: Выполнение\_услуги/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\_услуги,Срок\_выполнения,Номер\_песни,Номер\_договора,Код\_услуги,Стоимость")] Выполнение\_услуги выполнение\_услуги)

{

if (ModelState.IsValid)

{

try

{

db.Выполнение\_услуги.Add(выполнение\_услуги);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

ViewBag.Номер\_договора = new SelectList(db.Договоры, "Номер\_договора", "Паспорт\_заказчика", выполнение\_услуги.Номер\_договора);

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", выполнение\_услуги.Код\_услуги);

ViewBag.Номер\_песни = new SelectList(db.Песни, "Номер\_песни", "Название\_песни", выполнение\_услуги.Номер\_песни);

return View(выполнение\_услуги);

}

// GET: Выполнение\_услуги/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Выполнение\_услуги выполнение\_услуги = db.Выполнение\_услуги.Find(id);

if (выполнение\_услуги == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Номер\_договора = new SelectList(db.Договоры, "Номер\_договора", "Паспорт\_заказчика", выполнение\_услуги.Номер\_договора);

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", выполнение\_услуги.Код\_услуги);

ViewBag.Номер\_песни = new SelectList(db.Песни, "Номер\_песни", "Название\_песни", выполнение\_услуги.Номер\_песни);

return View(выполнение\_услуги);

}

// POST: Выполнение\_услуги/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\_услуги,Срок\_выполнения,Номер\_песни,Номер\_договора,Код\_услуги,Стоимость")] Выполнение\_услуги выполнение\_услуги)

{

if (ModelState.IsValid)

{

try{db.Entry(выполнение\_услуги).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Номер\_договора = new SelectList(db.Договоры, "Номер\_договора", "Паспорт\_заказчика", выполнение\_услуги.Номер\_договора);

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", выполнение\_услуги.Код\_услуги);

ViewBag.Номер\_песни = new SelectList(db.Песни, "Номер\_песни", "Название\_песни", выполнение\_услуги.Номер\_песни);

return View(выполнение\_услуги);

}

// GET: Выполнение\_услуги/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Выполнение\_услуги выполнение\_услуги = db.Выполнение\_услуги.Find(id);

if (выполнение\_услуги == null)

{

return RedirectToAction("Error404", "Error");

}

return View(выполнение\_услуги);

}

// POST: Выполнение\_услуги/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

try{Выполнение\_услуги выполнение\_услуги = db.Выполнение\_услуги.Find(id);

db.Выполнение\_услуги.Remove(выполнение\_услуги);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**ДоговорыController.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Data;

using System.Data.Entity;

using System.Diagnostics;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using KourseWork.Models;

using System.Data.SqlClient;

namespace KoursWork.Controllers

{

public class ДоговорыController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Договоры

public ActionResult Index()

{

var договоры = db.Договоры.Include(д => д.Исполнитель);

return View(договоры.ToList());

}

[HttpPost]

public ActionResult Index(string DogovorNumber, string period\_1, string period\_2)

{

var договоры = db.Договоры.Include(д => д.Исполнитель);

try

{

if (!string.IsNullOrEmpty(DogovorNumber))

{

using (AudioStudioDBEntities2 db = new AudioStudioDBEntities2())

{

System.Data.SqlClient.SqlParameter parameter = new System.Data.SqlClient.SqlParameter("@Dog\_count", DogovorNumber);

db.Database.ExecuteSqlCommand("Dogovor @Dog\_count", parameter);

}

}

}

catch

{

ViewBag.Data = "Пустой договор!";

return RedirectToAction("Error");

}

if(!string.IsNullOrEmpty(period\_1) && (!string.IsNullOrEmpty(period\_2)))

{

decimal? rat = db.PeriodSell(DateTime.Parse(period\_1), DateTime.Parse(period\_2)).First().Salary;

ViewBag.Result = "Выручка за данный период: " + rat + ". ";

return View(договоры.ToList());

}

return View(договоры.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Договоры/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Договоры договоры = db.Договоры.Find(id);

if (договоры == null)

{

return RedirectToAction("Error404", "Error");

}

return View(договоры);

}

// GET: Договоры/Create

public ActionResult Create()

{

ViewBag.Паспорт\_заказчика = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО");

return View();

}

// POST: Договоры/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Номер\_договора,Стоимость,Дата\_заключения,Кол\_во\_услуг,Паспорт\_заказчика")] Договоры договоры)

{

if (ModelState.IsValid)

{

try

{

db.Договоры.Add(договоры);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка создания пользователя.";

return RedirectToAction("Error");

}

}

ViewBag.Паспорт\_заказчика = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", договоры.Паспорт\_заказчика);

return View(договоры);

}

// GET: Договоры/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Договоры договоры = db.Договоры.Find(id);

if (договоры == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Паспорт\_заказчика = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", договоры.Паспорт\_заказчика);

return View(договоры);

}

// POST: Договоры/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Номер\_договора,Стоимость,Дата\_заключения,Кол\_во\_услуг,Паспорт\_заказчика")] Договоры договоры)

{

if (ModelState.IsValid)

{

try{ db.Entry(договоры).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Паспорт\_заказчика = new SelectList(db.Исполнитель, "Паспорт\_Исполнителя", "ФИО", договоры.Паспорт\_заказчика);

return View(договоры);

}

// GET: Договоры/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Договоры договоры = db.Договоры.Find(id);

if (договоры == null)

{

return RedirectToAction("Error404", "Error");

}

return View(договоры);

}

// POST: Договоры/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{Договоры договоры = db.Договоры.Find(id);

db.Договоры.Remove(договоры);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**ЗаписьController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class ЗаписьController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Запись

public ActionResult Index()

{

var запись = db.Запись.Include(з => з.Перечень\_услуг);

return View(запись.ToList());

}

// GET: Запись/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Запись запись = db.Запись.Find(id);

if (запись == null)

{

return RedirectToAction("Error404", "Error");

}

return View(запись);

}

// GET: Запись/Create

public ActionResult Create()

{

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование");

return View();

}

// POST: Запись/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 = "Код\_услуги,Перезаписи,Фрагменты,Что\_записывается,Сопр\_\_инструменты,Битрейт")] Запись запись)

{

if (ModelState.IsValid)

{

try{db.Запись.Add(запись);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка создания";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", запись.Код\_услуги);

return View(запись);

}

// GET: Запись/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Запись запись = db.Запись.Find(id);

if (запись == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", запись.Код\_услуги);

return View(запись);

}

// POST: Запись/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 = "Код\_услуги,Перезаписи,Фрагменты,Что\_записывается,Сопр\_\_инструменты,Битрейт")] Запись запись)

{

if (ModelState.IsValid)

{

try{db.Entry(запись).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактировани!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", запись.Код\_услуги);

return View(запись);

}

// GET: Запись/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Запись запись = db.Запись.Find(id);

if (запись == null)

{

return RedirectToAction("Error404", "Error");

}

return View(запись);

}

// POST: Запись/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{Запись запись = db.Запись.Find(id);

db.Запись.Remove(запись);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**ИсполнительController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class ИсполнительController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Исполнитель

public ActionResult Index()

{

return View(db.Исполнитель.ToList());

}

public ActionResult Error()

{

return View();

}

public ActionResult UserIndex()

{

return View(db.Исполнитель.ToList());

}

public ActionResult GuestIndex()

{

return View(db.Исполнитель.ToList());

}

// GET: Исполнитель/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Исполнитель исполнитель = db.Исполнитель.Find(id);

if (исполнитель == null)

{

return RedirectToAction("Error404", "Error");

}

return View(исполнитель);

}

// GET: Исполнитель/Create

public ActionResult Create()

{

return View();

}

// POST: Исполнитель/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 = "Паспорт\_Исполнителя,ФИО,Кол\_во\_альбомов,Карьера,Кол\_во\_наград,Страна,Эл\_\_почта")] Исполнитель исполнитель)

{

if (ModelState.IsValid)

{

try

{

db.Исполнитель.Add(исполнитель);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

return View(исполнитель);

}

// GET: Исполнитель/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Исполнитель исполнитель = db.Исполнитель.Find(id);

if (исполнитель == null)

{

return RedirectToAction("Error404", "Error");

}

return View(исполнитель);

}

// POST: Исполнитель/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 = "Паспорт\_Исполнителя,ФИО,Кол\_во\_альбомов,Карьера,Кол\_во\_наград,Страна,Эл\_\_почта")] Исполнитель исполнитель)

{

if (ModelState.IsValid)

{

try{db.Entry(исполнитель).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

return View(исполнитель);

}

// GET: Исполнитель/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Исполнитель исполнитель = db.Исполнитель.Find(id);

if (исполнитель == null)

{

return RedirectToAction("Error404", "Error");

}

return View(исполнитель);

}

// POST: Исполнитель/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{

Исполнитель исполнитель = db.Исполнитель.Find(id);

db.Исполнитель.Remove(исполнитель);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**Перечень\_услугController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class Перечень\_услугController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Перечень\_услуг

public ActionResult Index()

{

var перечень\_услуг = db.Перечень\_услуг.Include(п => п.Аранжировка).Include(п => п.Запись).Include(п => п.Сотрудники).Include(п => п.Сведение);

return View(перечень\_услуг.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Перечень\_услуг/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Перечень\_услуг перечень\_услуг = db.Перечень\_услуг.Find(id);

if (перечень\_услуг == null)

{

return RedirectToAction("Error404", "Error");

}

return View(перечень\_услуг);

}

// GET: Перечень\_услуг/Create

public ActionResult Create()

{

ViewBag.Код\_услуги = new SelectList(db.Аранжировка, "Код\_услуги", "Референс");

ViewBag.Код\_услуги = new SelectList(db.Запись, "Код\_услуги", "Что\_записывается");

ViewBag.Паспорт\_сотрудника = new SelectList(db.Сотрудники, "Паспорт\_сотрудника", "ФИО");

ViewBag.Код\_услуги = new SelectList(db.Сведение, "Код\_услуги", "Плагин");

return View();

}

// POST: Перечень\_услуг/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 = "Код\_услуги,Оборудование,Название,Паспорт\_сотрудника")] Перечень\_услуг перечень\_услуг)

{

if (ModelState.IsValid)

{

try

{

db.Перечень\_услуг.Add(перечень\_услуг);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Аранжировка, "Код\_услуги", "Референс", перечень\_услуг.Код\_услуги);

ViewBag.Код\_услуги = new SelectList(db.Запись, "Код\_услуги", "Что\_записывается", перечень\_услуг.Код\_услуги);

ViewBag.Паспорт\_сотрудника = new SelectList(db.Сотрудники, "Паспорт\_сотрудника", "ФИО", перечень\_услуг.Паспорт\_сотрудника);

ViewBag.Код\_услуги = new SelectList(db.Сведение, "Код\_услуги", "Плагин", перечень\_услуг.Код\_услуги);

return View(перечень\_услуг);

}

// GET: Перечень\_услуг/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Перечень\_услуг перечень\_услуг = db.Перечень\_услуг.Find(id);

if (перечень\_услуг == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Код\_услуги = new SelectList(db.Аранжировка, "Код\_услуги", "Референс", перечень\_услуг.Код\_услуги);

ViewBag.Код\_услуги = new SelectList(db.Запись, "Код\_услуги", "Что\_записывается", перечень\_услуг.Код\_услуги);

ViewBag.Паспорт\_сотрудника = new SelectList(db.Сотрудники, "Паспорт\_сотрудника", "ФИО", перечень\_услуг.Паспорт\_сотрудника);

ViewBag.Код\_услуги = new SelectList(db.Сведение, "Код\_услуги", "Плагин", перечень\_услуг.Код\_услуги);

return View(перечень\_услуг);

}

// POST: Перечень\_услуг/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 = "Код\_услуги,Оборудование,Название,Паспорт\_сотрудника")] Перечень\_услуг перечень\_услуг)

{

if (ModelState.IsValid)

{

try{db.Entry(перечень\_услуг).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Аранжировка, "Код\_услуги", "Референс", перечень\_услуг.Код\_услуги);

ViewBag.Код\_услуги = new SelectList(db.Запись, "Код\_услуги", "Что\_записывается", перечень\_услуг.Код\_услуги);

ViewBag.Паспорт\_сотрудника = new SelectList(db.Сотрудники, "Паспорт\_сотрудника", "ФИО", перечень\_услуг.Паспорт\_сотрудника);

ViewBag.Код\_услуги = new SelectList(db.Сведение, "Код\_услуги", "Плагин", перечень\_услуг.Код\_услуги);

return View(перечень\_услуг);

}

// GET: Перечень\_услуг/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Перечень\_услуг перечень\_услуг = db.Перечень\_услуг.Find(id);

if (перечень\_услуг == null)

{

return RedirectToAction("Error404", "Error");

}

return View(перечень\_услуг);

}

// POST: Перечень\_услуг/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{Перечень\_услуг перечень\_услуг = db.Перечень\_услуг.Find(id);

db.Перечень\_услуг.Remove(перечень\_услуг);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**ПесниController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class ПесниController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Песни

public ActionResult Index()

{

var песни = db.Песни.Include(п => п.Альбомы);

return View(песни.ToList());

}

public ActionResult UserIndex()

{

var песни = db.Песни.Include(п => п.Альбомы);

return View(песни.ToList());

}

public ActionResult GuestIndex()

{

var песни = db.Песни.Include(п => п.Альбомы);

return View(песни.ToList());

}

[HttpPost]

public ActionResult Index(string SongName)

{

IEnumerable<Песни> AllSongs = db.Песни.AsEnumerable();

if (SongName != null)

{

AllSongs = db.Песни.Where(s => s.Название\_песни == SongName);

}

return View(AllSongs.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Песни/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Песни песни = db.Песни.Find(id);

if (песни == null)

{

return RedirectToAction("Error404", "Error");

}

return View(песни);

}

// GET: Песни/Create

public ActionResult Create()

{

ViewBag.Номер\_альбома = new SelectList(db.Альбомы, "Номер\_альбома", "Название\_альбома");

return View();

}

// POST: Песни/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\_песни,Название\_песни,Битрейт,Длина,Номер\_альбома,Сопр\_\_инструменты")] Песни песни)

{

if (ModelState.IsValid)

{

try

{

db.Песни.Add(песни);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

ViewBag.Номер\_альбома = new SelectList(db.Альбомы, "Номер\_альбома", "Название\_альбома", песни.Номер\_альбома);

return View(песни);

}

// GET: Песни/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Песни песни = db.Песни.Find(id);

if (песни == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Номер\_альбома = new SelectList(db.Альбомы, "Номер\_альбома", "Название\_альбома", песни.Номер\_альбома);

return View(песни);

}

// POST: Песни/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\_песни,Название\_песни,Битрейт,Длина,Номер\_альбома,Сопр\_\_инструменты")] Песни песни)

{

if (ModelState.IsValid)

{

try{db.Entry(песни).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

ViewBag.Номер\_альбома = new SelectList(db.Альбомы, "Номер\_альбома", "Название\_альбома", песни.Номер\_альбома);

return View(песни);

}

// GET: Песни/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Песни песни = db.Песни.Find(id);

if (песни == null)

{

return RedirectToAction("Error404", "Error");

}

return View(песни);

}

// POST: Песни/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Песни песни = db.Песни.Find(id);

db.Песни.Remove(песни);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**СведениеController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class СведениеController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Сведение

public ActionResult Index()

{

var сведение = db.Сведение.Include(с => с.Перечень\_услуг);

return View(сведение.ToList());

}

// GET: Сведение/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сведение сведение = db.Сведение.Find(id);

if (сведение == null)

{

return RedirectToAction("Error404", "Error");

}

return View(сведение);

}

// GET: Сведение/Create

public ActionResult Create()

{

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование");

return View();

}

// POST: Сведение/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 = "Код\_услуги,Кол\_во\_дорожек,Работа\_в\_днях,Плагин")] Сведение сведение)

{

if (ModelState.IsValid)

{

try{db.Сведение.Add(сведение);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка создания!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", сведение.Код\_услуги);

return View(сведение);

}

// GET: Сведение/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сведение сведение = db.Сведение.Find(id);

if (сведение == null)

{

return RedirectToAction("Error404", "Error");

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", сведение.Код\_услуги);

return View(сведение);

}

// POST: Сведение/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 = "Код\_услуги,Кол\_во\_дорожек,Работа\_в\_днях,Плагин")] Сведение сведение)

{

if (ModelState.IsValid)

{

try{db.Entry(сведение).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка изменения!";

return RedirectToAction("Error");

}

}

ViewBag.Код\_услуги = new SelectList(db.Перечень\_услуг, "Код\_услуги", "Оборудование", сведение.Код\_услуги);

return View(сведение);

}

// GET: Сведение/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сведение сведение = db.Сведение.Find(id);

if (сведение == null)

{

return RedirectToAction("Error404", "Error");

}

return View(сведение);

}

// POST: Сведение/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{Сведение сведение = db.Сведение.Find(id);

db.Сведение.Remove(сведение);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**СотрудникиController.cs**

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 KourseWork.Models;

namespace KoursWork.Controllers

{

public class СотрудникиController : Controller

{

private AudioStudioDBEntities2 db = new AudioStudioDBEntities2();

// GET: Сотрудники

public ActionResult Index()

{

return View(db.Сотрудники.ToList());

}

[HttpPost]

public ActionResult Index(string Doljnost, string percent)

{

IEnumerable<Сотрудники> сотрудники = db.Сотрудники.AsEnumerable();

if (Doljnost != null)

{

сотрудники = db.Сотрудники.Where(x => x.Должность == Doljnost);

}

try

{

if (!string.IsNullOrEmpty(percent))

{

int per = Convert.ToInt32(percent);

using (AudioStudioDBEntities2 db = new AudioStudioDBEntities2())

{

System.Data.SqlClient.SqlParameter parameter = new System.Data.SqlClient.SqlParameter("@PerCent", per);

db.Database.ExecuteSqlCommand("Nalog @PerCent", parameter);

}

}

}

catch

{

ViewBag.Data = "Ошибка!";

return RedirectToAction("Error");

}

return View(сотрудники.ToList());

}

public ActionResult UserIndex(string Doljnost, string percent)

{

IEnumerable<Сотрудники> сотрудники = db.Сотрудники.AsEnumerable();

if (Doljnost != null)

{

сотрудники = db.Сотрудники.Where(x => x.Должность == Doljnost);

}

try

{

if (!string.IsNullOrEmpty(percent))

{

int per = Convert.ToInt32(percent);

using (AudioStudioDBEntities2 db = new AudioStudioDBEntities2())

{

System.Data.SqlClient.SqlParameter parameter = new System.Data.SqlClient.SqlParameter("@PerCent", per);

db.Database.ExecuteSqlCommand("Nalog @PerCent", parameter);

}

}

}

catch

{

ViewBag.Data = "Ошибка!";

return RedirectToAction("Error");

}

return View(сотрудники.ToList());

}

public ActionResult Error()

{

return View();

}

// GET: Сотрудники/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сотрудники сотрудники = db.Сотрудники.Find(id);

if (сотрудники == null)

{

return RedirectToAction("Error404", "Error");

}

return View(сотрудники);

}

// GET: Сотрудники/Create

public ActionResult Create()

{

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 = "Паспорт\_сотрудника,ФИО,Должность,Дата\_рождения,Телефон,Пол,Зарплата,Номер\_труд\_\_книги")] Сотрудники сотрудники)

{

if (ModelState.IsValid)

{

try

{

db.Сотрудники.Add(сотрудники);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Error!";

return RedirectToAction("Error");

}

}

return View(сотрудники);

}

// GET: Сотрудники/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сотрудники сотрудники = db.Сотрудники.Find(id);

if (сотрудники == null)

{

return RedirectToAction("Error404", "Error");

}

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 = "Паспорт\_сотрудника,ФИО,Должность,Дата\_рождения,Телефон,Пол,Зарплата,Номер\_труд\_\_книги")] Сотрудники сотрудники)

{

if (ModelState.IsValid)

{

try{

db.Entry(сотрудники).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка редактирования!";

return RedirectToAction("Error");

}

}

return View(сотрудники);

}

// GET: Сотрудники/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return RedirectToAction("Error404", "Error");

}

Сотрудники сотрудники = db.Сотрудники.Find(id);

if (сотрудники == null)

{

return RedirectToAction("Error404", "Error");

}

return View(сотрудники);

}

// POST: Сотрудники/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

try{Сотрудники сотрудники = db.Сотрудники.Find(id);

db.Сотрудники.Remove(сотрудники);

db.SaveChanges();

return RedirectToAction("Index");

}

catch

{

ViewBag.Data = "Ошибка удаления! Возможно, эти данные где-то есть...";

return RedirectToAction("Error");

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}