**СОДЕРЖАНИЕ**

[Введение 4](#_Toc153877309)

[1 Логическая и физическая структура базы данных 6](#_Toc153877310)

[1.1 Информационно-логическая модель информационной системы 6](#_Toc153877311)

[1.2 Физическая модель базы данных 10](#_Toc153877312)

[1.3 Файловая структура базы данных 12](#_Toc153877313)

[2 Аппаратное и программное обеспечение информа- ционной системы 13](#_Toc153877314)

[2.1 Требования к системному и прикладному программному обеспечению 13](#_Toc153877315)

[2.2 Настройка и развёртывание приложения на сервере 13](#_Toc153877316)

[3 Структура приложения 15](#_Toc153877317)

[3.1 Описание общей структуры веб-приложения 15](#_Toc153877318)

[3.2 Описание классов для доступа к данным 15](#_Toc153877319)

[3.3 Описание контроллеров 16](#_Toc153877320)

[3.4 Описание представлений 18](#_Toc153877321)

[4 Руководство пользователя 21](#_Toc153877322)

[4.1 Назначение, условие применения и функционал 21](#_Toc153877323)

[4.2 Подготовка к работе 21](#_Toc153877324)

[4.3 Описание операции по обработки данных 22](#_Toc153877325)

[5 Руководство программиста 25](#_Toc153877326)

[5.1 Назначения и условия применения программы 25](#_Toc153877327)

[5.2 Характеристики программы 25](#_Toc153877328)

[5.3 Сопровождение программного комплекса 25](#_Toc153877329)

[5.4 Входные и выходные данные 26](#_Toc153877330)

[5.5 Сообщения в ходе работы приложения 26](#_Toc153877331)

[Заключение 27](#_Toc153877332)

[Список используемых источников 28](#_Toc153877333)

[Приложение А 29](#_Toc153877334)

[Приложение Б 134](#_Toc153877335)

# ВВЕДЕНИЕ

Тематикой курсового проекта является разработка *web*-приложения базы данных сервиса по прокату дисков и видеокассет, создание интерфейса в виде набора *web*-страниц, обеспечивающих отображение и редактирование информации из базы данных, для автоматизации работы со структурированной информацией.

Наиболее популярным типом информационных архитектур является клиент-серверная архитектура. Эта архитектура представляет собой взаимодействие структурных компонентов, которыми являются сервер и узлы-поставщики определённых сервисов, а также клиенты, которые пользуются данным сервисом. Такой тип архитектуры наиболее часто используется в создании корпоративных информационных систем, в которых база данных является главным элементом, а все необходимые операции с базой выполняются сервером. Запросы на получение и изменение информации из базы данных отправляют клиенты. Сервер обрабатывает запросы и возвращает ответ клиенту. Преимуществом такой системы является её достаточно высокий уровень производительности за счёт распределения вычислительной нагрузки между клиентом и сервером, а также непротиворечивость данных за счёт централизованной обработки.

Задачей курсового проекта является проектирование и создание базы данных в выбранной СУБД и разработка веб-приложения, которое обеспечивает отображение, редактирование и обработку информации из разработанной базы данных. Структура базы данных должна быть нормализована – таблицы базы данных должны удовлетворять требованиям третьей нормальной формы. База данных должна содержать тестовый набор данных (не менее 100 записей у таблицы на стороне отношения «один» и не менее 10000 записей у таблицы на стороне отношения «многие»).

Для поставленной задачи определены следующие исходные данные:

– жанры фильмов (наименование, описание);

– диски (наименование и год создания фильма, производитель, страна, главный актёр, дата записи, жанр, тип диска, цена);

– клиенты (ФИО, адрес, телефон, паспортные данные, дата взятия, дата возврата, отметка об оплате, отметка о возврате, взятые диски – до трех, сотрудник);

Также прилагаются следующие дополнительные требования к отображению данных:

– отдел кадров (данные обо всех сотрудниках и их должностях, отображение сведений о сотрудниках с определенными: должностью, стажем работы в данной организации);

– диски и жанры (отображение полных сведений обо всех дисках; сведений о дисках: с фильмами определенных жанров, фильмами с определенным главным актером);

– диски на руках (отображение полных сведений о всех выданных дисках; сведений о всех выданных дисках с определенным фильмом);

– полные сведения о дисках, не сданных вовремя.

Для решения поставленной задачи в качестве СУБД используется *MS SQL Server*. Эта СУБД обеспечивает поддержку баз данных очень большого объёма и обработку сложных запросов, а также имеет эффективные алгоритмы для работы с памятью и автоматизированным контролем размера файлов баз данных. В качестве технологии для разработки веб-приложения используется платформа *ASP.NET Core MVC* [3, с. 105]. Данная платформа является многофункциональной платформой для создания веб-приложений с помощью шаблона проектирования *Model-View-Controller* (модель-контроллер-представление). Структура *MVC* предполагает разделение приложения на три основных компонента: модель, представление и контроллер [6]. Каждый компонент решает свои задачи и взаимодействует с другими компонентами. Т.е. данный шаблон проектирования позволяет разделить задачи для каждого компонента, позволяет разрабатывать проект в команде, разделяя задачи между участниками и обеспечивает дальнейшую масштабируемость проекта. Благодаря такой схеме связей и распределения обязанностей между компонентами процесс масштабирования приложения становится проще, т.к. облегчается процесс написания кода, выполнения отладки и тестирования компонентов. Для доступа к данным используется технология *Entity Framework Core*. Эта технология является *ORM (object-relational mapping –* отображение данных на реальные объекты) инструментом, т.е. она позволяет работать с реляционными данными, используя классы и их иерархии. Также основным преимуществом данной технологии является использование универсального интерфейса для работы с данными, что позволяет легко и быстро сменить СУБД.

# 1 ЛОГИЧЕСКАЯ И ФИЗИЧЕСКАЯ СТРУКТУРА БАЗЫ ДАННЫХ

## **1.1 Информационно-логическая модель информационной системы**

Для решения задачи была сформирована структура и логика приложения. В первую очередь из исходных данных были выделены следующие сущности:

– Клиенты;

– Диски;

– Жанры;

– Должности;

– Список цен;

– Производители;

– Сотрудники;

– Информация о взятых дисках;

– Типы дисков.

Для сущности «Клиенты» было создано отношение (таблица) с атрибутами: «Код клиента», «Фамилия», «Имя», «Отчество», «Адрес», «Номер телефона», «Номер паспорта». Подробное описание отношения и атрибутов приведено в таблице 1.1. Данное отношение находится в первой нормальной форме.

Таблица 1.1 – Структура сущности «Клиенты»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код клиента | Уникальный инкрементируемый идентификатор для каждого клиента. Является первичным ключом. | Целое число |
| Фамилия | Содержит фамилию клиента. | Строка |
| Имя | Содержит имя клиента. | Строка |
| Отчество | Содержит отчество клиента. | Строка |
| Адрес | Содержит адрес проживания клиента. | Строка |
| Номер телефона | Содержит номер телефона клиента. | Строка |
| Номер паспорта | Содержит номер паспорта клиента. | Строка |

Для сущности «Жанры» было создано отношение (таблица) с атрибутами: «Код жанра», «Название», «Описание. Подробное описание отношения и атрибутов приведено в таблице 1.2. Данное отношение находится в первой нормальной форме.

Таблица 1.2 – Структура сущности «Жанры»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код жанра | Уникальный инкрементируемый идентификатор для каждого жанра. Является первичным ключом. | Целое число |
| Название | Содержит название жанра. | Строка |
| Описание | Содержит описание жанра. | Строка |

Для сущности «Должности» было создано отношение (таблица) с атрибутами: «Код должности», «Название». Подробное описание отношения и атрибутов приведено в таблице 1.3. Данное отношение находится в первой нормальной форме.

Таблица 1.3 – Структура сущности «Должности»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код должности | Уникальный инкрементируемый идентификатор для каждой должности. Является первичным ключом. | Целое число |
| Название | Содержит название должности. | Строка |

Для сущности «Производители» было создано отношение (таблица) с атрибутами: «Код производителя», «Название», «Страна». Подробное описание отношения и атрибутов приведено в таблице 1.4. Данное отношение находится в первой нормальной форме.

Таблица 1.4 – Структура сущности «Производители»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код производителя | Уникальный инкрементируемый идентификатор для каждого производителя. Является первичным ключом. | Целое число |
| Название | Содержит название производителя. | Строка |
| Страна | Содержит название страны производителя. | Строка |

Для сущности «Типы» было создано отношение (таблица) с атрибутами: «Код типа», «Название», «Описание». Подробное описание отношения и атрибутов приведено в таблице 1.5. Данное отношение находится в первой нормальной форме.

Таблица 1.5 – Структура сущности «Типы»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код типа | Уникальный инкрементируемый идентификатор для каждого типа. Является первичным ключом. | Целое число |
| Название | Содержит название типа носителя информации. | Строка |
| Описание | Содержит описание носителя информации. | Строка |

Отношение для сущности «Диски», описано в таблице 1.6. Отношение по условию задачи должно содержать атрибуты: «Код диска», «Название фильма», «Год создания фильма», «Производитель», «Страна», «Главный актёр», «Дата записи», «Жанр», «Тип диска», «Цена». Данное отношение следует привести к третей нормальной форме, заменив атрибуты «Производитель» и «Страна» на атрибут «Код производителя» связав отношение «Диски» с отношением «Производители», атрибут «Жанр» на атрибут «Код жанра», связав отношение «Диски» с отношением «Жанры», атрибут «Тип диска» на атрибут «Код типа», связав отношение «Диски» с отношением «Типа», а атрибут «Цена» вынесен в отношение «Список цен», ссылающееся на атрибут «Код диска», в отношении «Диски».

Таблица 1.6 – Структура сущности «Диски»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код диска | Уникальный инкрементируемый идентификатор для каждого диска. Является первичным ключом. | Целое число |
| Название | Содержит название фильма, записанного на диск. | Строка |
| Год создания | Содержит числовое значение года создания записанного фильма. | Строка |
| Код производителя | Содержит ссылки на код производителя. Является внешним ключом для связи с отношением «Производители» | Целое число |
| Главный актёр | Содержит данные о главном актёре в записанном на диск фильме. | Строка |
| Дата записи | Содержит дату записи фильма на носитель. | Строка |
| Код жанра | Содержит ссылки на код жанра. Является внешним ключом для связи с отношением «Жанры». | Целое число |
| Код типа | Содержит ссылки на код типа. Является внешним ключом для связи с отношением «Жанры». | Целое число |

Для сущности «Список цен» было создано отношение (таблица) с атрибутами: «Код цены», «Код диска», «Цена». Подробное описание отношения и атрибутов приведено в таблице 1.7. Данное отношение находится в третьей нормальной форме.

Таблица 1.7 – Структура сущности «Список цен»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код цены | Уникальный инкрементируемый идентификатор для каждой цены. Является первичным ключом. | Целое число |
| Код диска | Содержит код диска с фильмом. Является внешним ключом для связи с отношением «Диски». | Целое число |
| Цена | Содержит значение цены диска. | Деньги |

Для сущности «Сотрудники» было создано отношение (таблица) с атрибутами: «Код сотрудника», «Фамилия», «Имя», «Отчество», «Код должности», «Дата найма». Подробное описание отношения и атрибутов приведено в таблице 1.8. Данное отношение находится в третьей нормальной форме.

Таблица 1.8 – Структура сущности «Сотрудники»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код сотрудника | Уникальный инкрементируемый идентификатор для каждого сотрудника. Является первичным ключом. | Целое число |
| Фамилия | Содержит фамилию сотрудника. | Строка |
| Имя | Содержит имя сотрудника. | Строка |
| Отчество | Содержит отчество сотрудника. | Строка |
| Код должности | Содержит код должности. Является внешним ключом для связи с отношением «Должности». | Целое число |
| Дата найма | Содержит дату найма сотрудника | Дата |

Сущность «Информация о взятых дисках», таблица 1.9, состоит из атрибутов: «Код взятого диска», «Код клиента», «Код диска», «Дата взятия», «Дата возврата», «Отметка об оплате», «Отметка об возврате», «Код сотрудника».

Таблица 1.9 – Структура сущности «Информация о взятых дисках»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код взятого диска | Уникальный инкрементируемый идентификатор для каждого взятого диска. Является первичным ключом. | Целое число |
| Код клиента | Содержит код клиента. Является внешним ключом для связи с отношением «Клиенты». | Целое число |
| Код диска | Содержит код диска. Является внешним ключом для связи с отношением «Диски». | Целое число |
| Дата взятия | Содержит дату взятия диска. | Дата |
| Дата возврата | Содержит дату, до которой нужно вернуть диск. | Диск |
| Отметка об оплате | Содержит отметку об успешной оплате аренды диска. | Логический |
| Отметка об возврате | Содержит отметку о возврате диска. | Логический |
| Код сотрудника | Содержит код сотрудника. Является внешним ключом для связи с отношением «Сотрудники». | Целое число |

После определения всех отношений и атрибутов, тем самым была составлена информационно-логическая модель информационной системы.

## **1.2 Физическая модель базы данных**

По созданной информационно-логической модели была создана иерархия класса и контекст данных, приложение Б, которая описывает ранее созданные отношения атрибуты и домены, для каждого отношения был создан свой соответствующий класс и определены реляционные отношения между ими. Далее по подходу *Code First* с помощью средств *Entity Framework*, была сгенерирована база данных в СУБД *MS SQL Server*. После преобразования логической модели в физическую, в физической модели были получены таблицы со связями, соответствующими каждой из ранее определённых отношений, диаграмма базы данных и связи между сгенерированными таблицами представлены на рисунке 1.1.

На рисунке 1.1 изображена сгенерированная база данных с помощью *Entity Framework*.

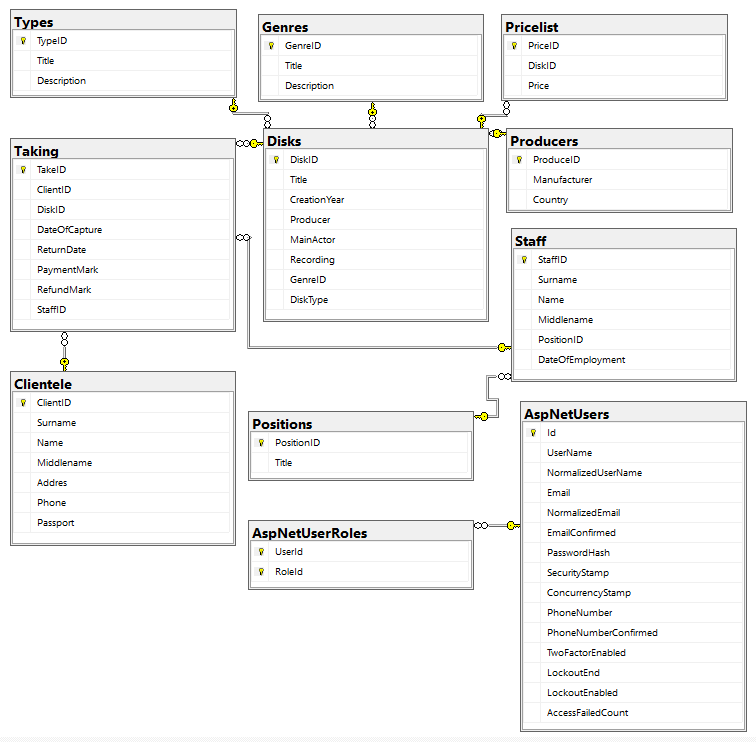


Рисунок 1.1 – Диаграмма базы данных

Для процесса преобразовании логической модели в физическую существует несколько правил:

– сущности становятся таблицами в физической базе данных;

– атрибуты становятся столбцами в физической базе данных. Также для каждого столбца необходимо определить подходящий тип данных;

– уникальные идентификаторы становятся столбцами, не допускающими значение *NULL*, т.е. первичными ключами. Также значение идентификатора делается автоинкрементным для обеспечения уникальности;

– все отношения моделируются в виде внешних ключей.

## **1.3 Файловая структура базы данных**

Все базы данных *MS SQL Server* имеют два основных рабочих системных файла: файл данных и файл журнала. Файлы данных содержат данные и объекты, такие как таблицы, индексы, хранимые процедуры и представления. Файлы журнала содержат сведения, необходимые для восстановления всех транзакций в базе данных.

Для каждого отношения были получены следующие таблицы: *Clientele, Staff, Positions, Producers, Taking, Genres, Pricelist, Disks, Types.*

Таблицы *Clientele*, *Positions*, *Producers, Genres, Types* находятся в отношении «один» и описывают сущности «Клиенты», «Должности», «Производители», «Жанры» и «Типы» и подобраны физические тип данных для соответствующих столбцов, установлены первичные ключи.

Таблицы *Taking*, *Staff*, *Disks*, *Pricelist* находятся в отношении «многие», описывают сущности «Взятые диски», «Сотрудники», «Диски» и «Список цен». Имеют автоинкрементируемый первичный ключ и внешние ключи для связи с таблицами в отношении «Один».

С помощью библиотеки *Entity Framework* было осуществлено взаимодействия языка программирования *C#* с физической моделью данных, который произвёл соотношения классов и таблиц, был создан контекст данных, с помощью которого можно осуществлять доступ непосредственно в коде приложения.

# 2 АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМА- ЦИОННОЙ СИСТЕМЫ

## **2.1 Требования к системному и прикладному программному обеспечению**

***2.1.1*** Для корректной работы аппаратного и программного обеспечения на стороне сервера хранилища данных, требуется соблюдения следующих условий:

– установленный *MS SQL Server*;

– для работы *MS SQL Server* 2016 и выше, требуется *.NET Framework 4.6*;

– сетевое программное обеспечение;

– требуется как минимум 7 ГБ свободного места на диске (при увеличении размера базы данных, может потребоваться свободного места);

– минимальный объем оперативной памяти 1 ГБ;

– процессор *x*64 с тактовой частотой 1,4 ГГц;

Требование перечисленные выше являются минимальными и могут меняться относительно размера базы данных и требуемых задач.

***2.1.2*** Минимальные требования к аппаратному и программному обеспечению на стороне *web*-сервера следующие:

– процессор *x*86/*x*64 с тактовой частотой 1 ГГц;

– минимальный объем оперативной памяти 512 МБ;

– требуется как минимум 4,5 ГБ свободного места на диске;

– операционные системы *Windows* 7, 8, 10, *Linux*, *Max OS*.

Так как приложение разработано на платформе *.NET*, оно является кроссплатформенным и может быть запущенно на любой поддерживаемой операционной системе. Для организации связи с СУБД требуется настроить подключение к нему. Так как СУБД может быть установлено на удалённом компьютере возможно потребуется подключение к интернету, либо к локальной сети, в которой находится сервер хранилища данных. Так же системные требования могут изменятся относительно масштаба приложения.

***2.1.3.*** Для того чтобы приложение корректно работало на стороне клиента требуется браузер с поддержкой «*Bootstrap*» и наличие клиента и *web*-сервера в одной сети (локальной, глобальной).

## **2.2 Настройка и развёртывание приложения на сервере**

Данное приложение может быть развёрнуто на серверах: *Apache Tomcat*, *Kestel*, *IIS*, *GlassFish* и др. Чтобы развернуть приложение, нужно перейти в папку с проектом и открыть командную строку и выполнить команду «*dotnet publish ProductOutputAndRelease -c Release*». После выполнении команды выходные данные приложения публикуется в папку «*./bin/Release/netcoreapp/publish*» относительно директории проекта.

Для запуска приложения веб-приложение нужно скопировать папку «*publish*» в директорию с установленным веб-сервером (в случаи Tomcat «*./webapp*») и выполнить команду «*dotnet ProductOutputAndRelease.dll*» с командной строки, после этого веб-приложение будет запущенно на сервере. Чтобы пользователь мог использовать веб-приложение, он должен находится в одной сети с веб-сервером.

Для подключения к базе данных, требуется сконфигурировать подключение к ней. Для этого требуется отредактировать конфигурационный файл приложения «*appsetting.json*» и изменить строку подключение. Для того чтобы веб-приложению удалось установить соединение с базой данных, СУБД и веб-приложение должны находится в одной сети [4].

# 3 СТРУКТУРА ПРИЛОЖЕНИЯ

## **3.1 Описание общей структуры веб-приложения**

В состав разработанного веб-приложения входят три основных компонента: модель, представление и контроллер.

Модель представляет состояние приложения и бизнес-логику, непосредственно связанную с данными. Как правило, объекты моделей хранятся в базе данных. В архитектуре *MVC* модели представлены двумя основными типами: модели представлений, которые используются представлениями для отображения данных на веб*-*странице, и модели домена, описывающие логику управления данными. Модель содержит данные и хранит логику обработки этих данных, но не содержит логику взаимодействия с пользователем, т.е. с представлением.

Представление является графическим веб*-*интерфейсом, через который пользователь может взаимодействовать с приложением напрямую. Этот компонент содержит минимальную логику, которая связана с представлением данных.

Контроллер представляет центральный компонент архитектуры *MVC* для управления взаимодействием с пользователем, работы с моделью и выбора представления для отображения. Контроллер обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки введённых пользователем данных и логику формирования ответа пользователю. Контроллер является начальной отправной точкой в приложении и отвечает за выбор рабочих типов моделей и отображаемых представлений.

## **3.2 Описание классов для доступа к данным**

Для работы с таблицами базы данных в приложении необходимы классы, которые описывают каждую таблицу. В данных классах описываются поля таблиц в виде свойств и связи между таблицами в виде связей между классами.

Классы *Clientele*, *Disk*, *Genre*, *Position*, *Pricelist*, *Producer*, *Staff*, *Taking*, *Type* описывают таблицы *Clientele*, *Disks*, *Types*, *Genres*, *Positions*, *Pricelist*, *Staff*, *Takings* соответственно. Код данных классов представлен в приложении А.

Свойства в каждом классе описывают столбцы соответствующей таблицы. В классах, описывающих таблицы, которые находятся на стороне отношения «многие», содержат ссылку на объект класса, моделирующего таблицу, связанную внешним ключом.

Также в данных классах используются аннотации – специальные атрибуты, которые определяют различные правила для отображения свойств модели. Для задания параметров отображения свойства используется атрибут *Display*. Этот атрибут устанавливает заголовок свойства, который используется при отображении названия свойства в представлении. Для предоставления среде выполнения информации о типе свойства используется атрибут *DataType*. Также для проверки значений свойств применяются специальные атрибуты валидации – *Required,* *RegularExpression* и *Range.* Атрибут *Required* помечает, что свойство должно быть обязательно установлено. С помощью свойства *ErrorMessage* этого атрибута задаётся выводимое при валидации сообщение. Атрибут *RegularExpression* помечает, что значение свойства должно соответствовать указанному в этом атрибуте регулярному выражению. Атрибут *Range* определяет минимальное и максимальное ограничение для свойств с числовым типом данных. Аналогично атрибут *StringLength* определяет ограничения для свойств строкового типа.

## **3.3 Описание контроллеров**

Контроллер представляет обычный класс, который наследуется от абстрактного базового класса *Microsoft.AspNetCore.Mvc.Controller*. Именование контроллеров строго предопределено, т.е. имя контроллера обязательно должно иметь суффикс «*Controller*», а остальная часть считается названием контроллера.

Адрес, который обрабатывается контроллерами, представлен в виде паттерна *{controller=[ControllerName]}/{action=[MethodName]}*, где [*ControllerNa-me*] – название контроллера, [*MethodName*] – название метода контроллера.

Для работы с созданными моделями разработаны следующие контроллеры:

– *HomeController* – отвечает за вывод начальной страницы;

– *ClienteleController –* отвечает за работу с таблицей *Clientele;*

*– DisksController –* отвечает за работу с таблицей *Disks;*

*– TypesController –* отвечает за работу с таблицей *Types;*

*– GenresController –* отвечает за работу с таблицей *Genres;*

*– PricelistController –* отвечает за работу с таблицей *Pricelist;*

*– ProducersController –* отвечает за работу с таблицей *Producers;*

*– StaffController –* отвечает за работу с таблицей *Staff;*

*– TakingsController –* отвечает за работу с таблицей *Takings;*

*– PositionsController –* отвечает за работу с таблицей *Positions.*

Контроллеры, отвечающие за работу с таблицами, имеют следующие методы:

– *Index;*

*– Details[GET];*

*– Create[GET];*

*– Create[POST];*

*– Edit[GET];*

*– Edit[POST];*

*– Delete[GET];*

Метод *Index* в качестве входных параметров принимает значения, по которым производится фильтрация данных, флаг фильтра и номер страницы. Флаг фильтра указывает, являются ли входные значения фильтров новыми или нет. Если фильтры новые (т.е. они не применялись для фильтрации данных), то происходит выборка данных из базы данных, фильтрация с использованием входных значений фильтров, формирование ключа кеша и запись данных в кеш. Если входные фильтры использовались, то происходит формирование ключа кеша и получение данных из кеша по ключу. Сформированный ключ добавляется в список с ключами, а применяемые фильтры сохраняются в сессию. Данный метод возвращает объект класса *IndexViewModel<T>*, который содержит отфильтрованные данные, значения фильтров и объект класса *PageViewModel*, содержащий свойства и методы, необходимые для работы страничной навигации.

Метод *Details[GET]* принимает идентификатор записи, производит выборку нужной записи из определённой таблицы базы данных и возвращает объект, моделирующий эту таблицу и содержащий все данные из таблицы.

Метод *Create[GET]* возвращает одноимённое представление с полями для добавления записи в таблицу базы данных. Для таблиц, стоящих на стороне «многие» данный метод формирует словари *ViewData*, в которые добавляются необходимые данные из таблиц, стоящих на стороне отношения «один».

Метод *Create[POST]* вызывается при отправке результата формы создания записи. Данный метод принимает объект, таблицу которого он моделирует и содержит данные, которые необходимо записать в базу данных. Перед записью производится валидация данных. Если данные неверны, то формируется ошибка, которая выводится в представлении. Если данные верны, то происходит запись данных в базу и переход в метод *Index* текущего контроллера.

Метод *Edit[GET]* принимает идентификатор записи и производит выборку нужной записи из определённой таблицы базы данных. Если запись найдена, то происходит добавление необходимых данных из других таблиц в словари *ViewData* и возврат представления с формой редактирования записи. Если запись не найдена, то метод возвращает стандартное сообщение об ошибке.

Метод *Edit[POST]* вызывается при отправке результата формы редактирования записи. Данный метод в качестве входных параметров принимает идентификатор записи и объект, содержащий данные об этой записи. Если входной идентификатор и идентификатор объекта не совпадают, то метод возвращает стандартное сообщение об ошибке. Иначе метод выполняет валидацию входных данных и если данные верны, то производится обновление данных в базе. Если операция обновления прошла успешно, то происходит переход в метод *Index* текущего контроллера. В случае возникновения ошибки метод возвращает стандартное сообщение об ошибке.

## **3.4 Описание представлений**

Представления – это файлы в формате *cshtml*, в которых используется язык разметки *HTML* и язык программирования *C#* в разметке *Razor*. Все представления объединяются в папки с именами, соответствующими названиям контроллеров. Все эти папки находятся в папке *Views* в корне приложения.

Для существующих контроллеров разработаны представления, которые содержатся следующих в папках:

– *Account* – содержит представления для работы с пользователями:

– *Login* – представление с формой для входа пользователя в систему;

– *Logout* – представление для выхода пользователя из системы;

– *Register* – представление с формой для регистрации нового пользователя в системе;

– *Clientele –* содержит представления для работы с данными о клиентах:

– *Index* – представление со списком всех клиентов;

– *Create* – представление с формой для добавления нового клиента;

– *Delete* – представление для удаления пользователя;

– *Edit* – представление для редактирования данных пользователя;

– *Disks –* содержит представления для работы с данными о дисках:

– *Index* – представление со списком всех дисков;

– *Create* – представление с формой для добавления нового диска;

– *Delete* – представление для удаления дисков;

– *Edit* – представление для редактирования данных диска;

– *Genres* – содержит представления для работы с данными о жанрах фильмов:

– *Index* – представление со списком всех жанров;

– *Create* – представление с формой для добавления нового жанра;

– *Delete* – представление для удаления жанра;

– *Edit* – представление для редактирования данных жанра;

– *Positions –* содержит представления для работы с информацией о должностях работников:

– *Index* – представление со списком всех должностей;

– *Create* – представление с формой для добавления новой должности;

– *Delete* – представление для удаления должности;

– *Edit* – представление для редактирования данных должности;

– *Pricelist –* содержит представления для работы со списком цен на прокат дисков:

– *Index* – представление со списком всех цен;

– *Create* – представление с формой для добавления нового ценника;

– *Delete* – представление для удаления ценника;

– *Edit* – представление для редактирования ценника;

– *Producers –* содержит представления для работы с информацией о производителях дисков:

– *Index* – представление со списком всех производителей;

– *Create* – представление с формой для добавления нового производителя;

– *Delete* – представление для удаления производителя;

– *Edit* – представление для редактирования данных производителя;

– *Staff –* содержит представления для работы со списком сотрудников компании по видеопрокату:

– *Index* – представление со списком всех клиентов;

– *Create* – представление с формой для добавления нового клиента;

– *Delete* – представление для удаления пользователя;

– *Edit* – представление для редактирования данных пользователя;

– *Takings –* содержит представления для работы с информацией о взятых в прокат дисках:

– *Index* – представление со списком всех взятых дисков;

– *Create* – представление с формой для добавления взятого диска;

– *Delete* – представление для удаления взятого диска;

– *Edit* – представление для редактирования данных о взятом диске;

– *Types –* содержит представления для работы со списком типов видео-носителей:

– *Index* – представление со списком всех типов носителей;

– *Create* – представление с формой для добавления нового носителя;

– *Delete* – представление для удаления носителя;

– *Edit* – представление для редактирования данных носителя;

Для каждого представления с выборкой данных был разработан класс «модель-представление» (*ViewModel*), данный класс нужен для создание постраничной навигации. Так же эти классы содержат объекты для дополнительной манипуляции с данными (фильтрации и сортировки).

Существует набор данных, который редко изменяется, а именно:

– Жанры;

– Должности;

– Производители;

– Типы носителей;

– Сотрудники;

Эти данные кешируется в кеше браузера с помощью атрибута *ResponseCache* (кешируются *css* стили, *html* страничка) и с помощью интерфейса *IMemoryCache* (кешируются данные выборки). Для кеширования с помощью *IMemoryCache* были реализованы дополнительные классы, которые подключаются как «сервисы» и благодаря технологии «*Dependency Injection*» (внедрение зависимости) неявно передаются классом котроллерам [5]. Записи в остальных таблицах изменяются значительно чаще, поэтому их регулярное кеширование не является рациональным.

# 4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## **4.1 Назначение, условие применения и функционал**

*Web*-приложение предназначено для управления и учёта данных о выданных дисках и кассетах.

Основные функции приложения:

– создание и вход в учётные записи пользователей;

– добавление, просмотр и редактирование списка дисков;

– выборка, сортировка и фильтрация списка дисков по заданным критериям;

– добавление, просмотр и редактирования данных о жанрах фильмов;

– добавление, просмотр и редактирования данных о типах носителей данных;

– добавление, просмотр и редактирования данных о клиентах видеопроката;

– добавление, просмотр и редактирования данных о должностях в компании;

– добавление, просмотр и редактирования данных о ценах на диски;

– добавление, просмотр и редактирования данных о производителях носителей;

– добавление, просмотр и редактирования данных о сотрудниках видеопроката;

– добавление, просмотр и редактирования данных о взятых в аренду дисках;

– автоматизация рабочего процесса.

## **4.2 Подготовка к работе**

Для использования приложения требуется веб-браузер (*Mozilla Firefox*, *Chrome*, *Opera*, *Microsoft Edge* и пр.) в адресной строке веб-браузера необходимо ввести *URL*-адрес выданный системным администратором, при условии нахождения устройства в той же локальной сети, где находится *web*-сервер (если сервер находится в глобальной сети, то подключение к интернету). После входа на веб-страницу приложения пользователь должен войти в существующую учётную запись, либо зарегистрировать новую.

При регистрации учётной записи пользователю необходимо заполнить поля логина и пароля (два раза для подтверждения). Также накладываются определённые условия на создание пароля: он должен состоять из не менее шести символов, должны присутствовать хотя бы по одной заглавной и строчной букве, а также хотя бы один специальный знак. В случае некорректности заполнения данных пользователь получает сообщение об ошибке. При успешной регистрации пользователь автоматически перенаправляется на главную страницу приложения со списком всех доступных для данного пользователя таблиц.

При входе на сайт пользователь должен ввести логин и пароль, а также, при желании, выбрать отметку для запоминания пользователя. В случае неверного указания данных пользователю отображается сообщение об ошибке. Если пункт «Запомнить» был отмечен, то при следующем заходе пользователь уже будет авторизован. При успешном входе пользователь автоматически перенаправляется на главную страницу приложения со списком всех доступных для данного пользователя таблиц.

## **4.3 Описание операции по обработки данных**

Для операции просмотра данных о дисках, требуется выбрать вкладку «Диски» в вверху страницы в браузере, рисунок 4.1.

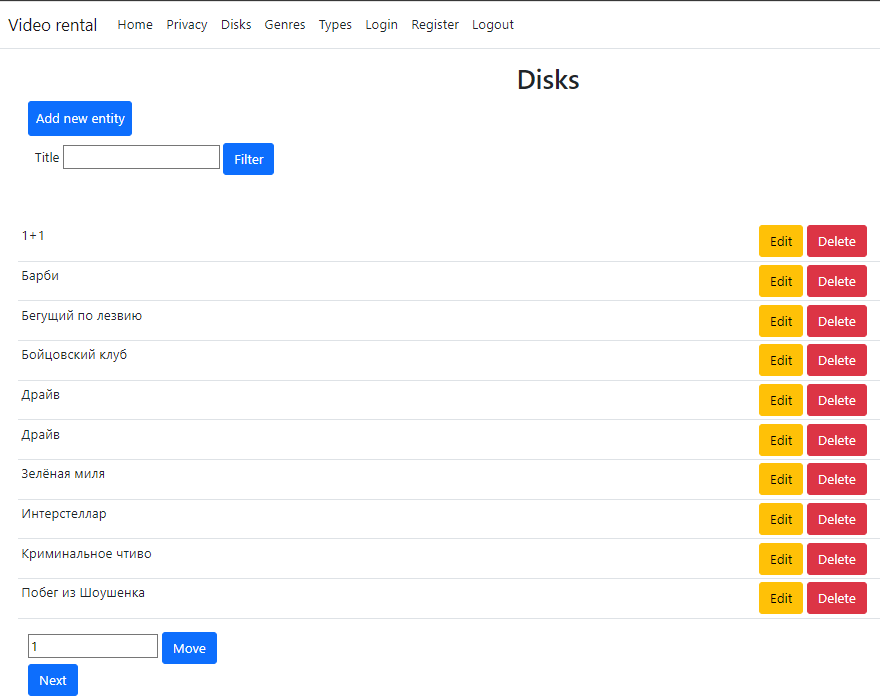


Рисунок 4.1 – Список дисков

На странице со писком данных в таблице реализованы методы сортировки записей по различным полям в алфавитном и обратном алфавитном порядке, по возрастанию и убыванию. Также реализован поиск диска по названию.

Чтобы найти фильм с определённым названием, искомое название надо ввести в строку поиска, рисунок 4.2.

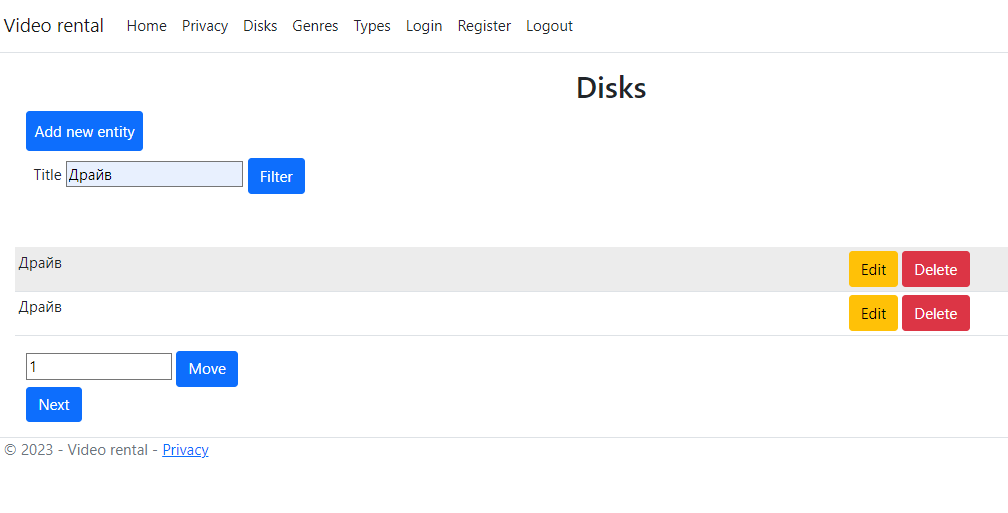


Рисунок 4.2 – Поиск дисков по названию

Данные из таблицы на странице отображаются не полностью, а разбиваются на страницы по 10 записей на каждой. Меню переключения страниц с записями реализовано внизу страницы. Можно перейти на конкретную страницу, введя номер искомой страницы, либо перемещаться с помощью кнопок «Следующая» и «Предыдущая».

На странице, отображающей записи из таблицы также имеются кнопки для добавления, удаления и изменения записей. Кнопка добавления вынесена отдельно в верхней части страницы, а кнопки редактирования и удаления есть у каждой записи в таблице. При нажатии на любую из них пользователь будет перенаправлен соответственно на страницы добавления, изменения или удаления записи.

В форме для добавления реализована валидация вводимых данных, а также удобный выбор даты с помощью открывающегося календаря, что не позволит ввести некорректные данные. При попытке ввода некорректных данных форма очистится и появится соответствующее сообщение.

На рисунке 4.3 представлена форма для добавления диска.

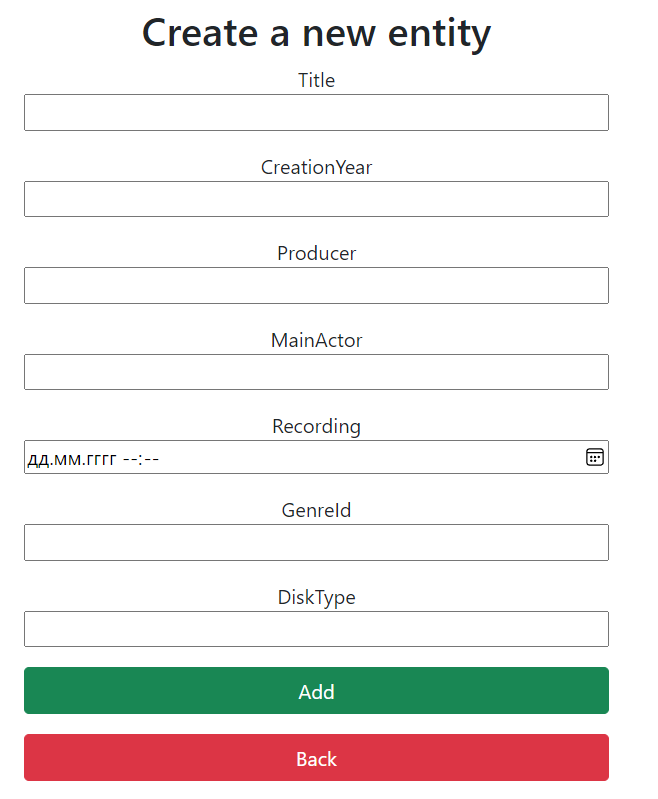


Рисунок 4.3 – Форма для добавления нового диска

После добавления диска пользователь будет автоматически перенаправлен на страницу со списком дисков.

Форма для изменения информации имеет такие же поля, функционал и внешний вид, как и форма для добавления записи.

На странице для удаления записи у пользователя потребуется подтверждение его действий, и если он нажмёт соответствующую кнопку, то будет перенаправлен на страницу со списком дисков, а запись будет удалена из таблицы.

Для других таблиц реализованы аналогичные страницы c таким же функционалом и внешним оформлением.

Если пользователь обладает ролью администратора, ему также будут доступны страницы со списком пользователей и ролевых политик. Администратор может добавить новую роль или выдать любому пользователю существующую роль.

# 5 РУКОВОДСТВО ПРОГРАММИСТА

## **5.1 Назначения и условия применения программы**

Приложение предназначена для предоставления информации из базы данных сервисам по прокату видеокассет и дисков, чтобы автоматизировать учёт заказов и прочие манипуляции с данными из базы.

Основные функции приложения:

– производить различные манипуляции с данными из базы данных;

– предоставления данных в удобном виде пользователям для их просмотра;

– управление данными отдела кадров;

– редактирования, добавление и изменения данных из базы с помощью веб-интерфейса.

Для запуска приложения на сервере должна быть установлена платформа *.NET Core*. Для соединения с базой данных, требуется предварительная конфигурация параметров для соединения с ней.

## **5.2 Характеристики программы**

Разработанное приложение написано на языке программирования *C#* в среде разработки *Visual Studio 2022*.

Для хранения данных используется база данных *MS SQL Server*. Работа с ней осуществляется с помощью библиотеки *Entity Framework*, работающая на основании стандартных драйверов для подключения *ADO*.

Серверная часть представляет собой *ASP.NET* приложение, к которому происходят запросы по протоколу *HTTP*, которые он обрабатывает и возвращает клиенту требуемую информацию. При работе используются следующие виды *HTTP*-глаголов: *GET*, *POST*.

## **5.3 Сопровождение программного комплекса**

Для дополнения программного обеспечения новым функционалом можно использовать любую среду разработки на языке программирования *C#*. Приложения реализовано с помощью паттерна *MVC* (*Model-View-Controller*), который позволяет в свою очередь разделить модель данных, бизнес-логику приложения и представления, на три части, что позволит разрабатывать новый функционал и поддерживать приложения в команде из нескольких разработчиков. Так же использование данного паттерна сделала приложение легко масштабируемым и поддерживаемым.

При необходимости можно заменить источник данных с *MS SQL Server* на другую базу данных, благодаря интерфейсу «источник данных».

## **5.4 Входные и выходные данные**

Входными данными для веб-приложения является:

– веб-сервер, на котором разворачивается приложение;

– сгенерированная база данных с помощью возможностей *Entity Framework*;

– тестовый набор для отладки приложения генерируемый компонентом Middleware (листинг приведён в приложении А).

Выходными данным для приложения является получение и предоставление данных с базы пользователю, их сортировка и выборка по критериям.

## **5.5 Сообщения в ходе работы приложения**

При работе программа может оповещать пользователя о следующих неполадках:

– некорректно введённые данных при добавлении и редактировании записей;

– некорректный *URL*-адрес, страница не найдена;

– ошибка при добавлении записей, запись с введёнными значениями уже существуют в базе.

Данные сообщения передаются в специальном виде ошибки с описанием проблемы.

# ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта было реализовано веб-приложение, которое производит автоматизацию работы сервиса по прокату видеокассет и дисков с фильмами. Приложение является простым и удобным благодаря адоптивному и понятному интерфейсу. Критериями удобства является в первую очередь наличие навигационного меню, что позволяет пользователю всю необходимую информацию, а также улучшает навигацию между страницами, не производя при этом никаких лишних действий.

Функционал приложения полностью выполняет поставленные задачи, и структура спроектирована таким образом, что его дальнейшее расширение не приведёт ни к каким трудностям: изменению структуры или переписыванию логики. Все вышеперечисленные преимущества, поможет небольшим сервисам видеопроката автоматизироваться свой рабочий процесс и учёт заказов.

В результате разработки курсового проекта, была изучена технология *ASP.NET Core MVC*.Технология позволяет использовать шаблоны, которые выполняют конкретные задачи. Так же благодаря платформе *.NET* приложение не зависит от операционной системы, или веб-сервера и является кроссплатформенной.

*MVC* описывает простой способ создания основной структуры приложения, что позволяет легко ориентироваться в коде, т.к. он разбит на блоки, а также серьёзно упрощает отладочный процесс.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Практическое руководство к курсовому проектированию по курсу «Информатика» для студентов технических специальностей дневной и заочной форм обучения – Гомель: ГГТУ им. П.О. Сухого, 2019. – 32 с.
2. Шилдт Герберт. C# 4.0: полное руководство: учебное пособие – ООО «И.Д. Вильямс», 2011. – 1056 с.
3. Чамберс Д., Пэкетт Д., Тиммс С., ASP.NET Core. Разработка приложений. – Спб.: Питер, 2018. – 464 с.
4. **Albahari J., Albahari B.,** C# 9.0 in a Nutshell – Sebastopol: O'Reilly Media, 2020. – 1064 с.
5. Griffiths I., Programming C# 8.0 - Beijing: O'Reilly Media, 2019. – 800 с.
6. Skeet J., C# in Depth - Shelter Island: Manning Publications, 2019. - 528 с.
7. **Stellman A., Greene** J., Head First C# - Beijing: O'Reilly Media, 2013. - 960 с.
8. **Troelsen A., Japikse P.,** Pro C# 9 with .NET 5 - Berkeley: Apress, 2021. - 1248 с.
9. **Michaelis M., Lippert E.,** Essential C# 8.0 - Redmond: Microsoft Press, 2019. - 1008 с.
10. Freeman A., Pro ASP.NET Core MVC 2 - Berkeley: Apress, 2017. - 1016 с.
11. Размещение и развёртывания ASP.NET Core приложения, – Электрон. данные. – Режим доступа: https://docs.microsoft.com/ru-ru/ASPNET/core/host-and-deploy/?view=aspnetcore-2.1. – Дата доступа: 12.12.2019.
12. ASP.NET Core. Dependency Injection, Электрон. данные. – Режим доступа: https://metanit.com/sharp/aspnet5/6.1.php. Дата доступа: 13.12.2019.
13. ASP.NET Core. Введение в MVC, Электрон. данные. – Режим доступа: https://metanit.com/sharp/aspnet5/3.1.php. Дата доступа: 13.12.2019.

# ПРИЛОЖЕНИЕ А

(обязательное)

**Код программы**

**Листинг файла *Program*.*cs*:**

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Services;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<VideoRentalContext>(options => options.UseSqlServer("Server=DESKTOP-I9HQNLA;Database=video\_rental;Trusted\_Connection=True;TrustServerCertificate=True"));

builder.Services.AddIdentity<User, IdentityRole>(opts =>

{

opts.Password.RequiredLength = 5; // минимальная длина

opts.Password.RequireNonAlphanumeric = false; // требуются ли не алфавитно-цифровые символы

opts.Password.RequireLowercase = false; // требуются ли символы в нижнем регистре

opts.Password.RequireUppercase = false; // требуются ли символы в верхнем регистре

opts.Password.RequireDigit = false; // требуются ли цифры

})

.AddEntityFrameworkStores<VideoRentalContext>();

builder.Services.AddTransient<CacheProvider>();

builder.Services.AddMemoryCache();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddMvc();

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie(options => options.LoginPath = "/Account/Login");

builder.Services.AddAuthorization();

builder.Services.AddSession(options =>

{

options.Cookie.Name = "VideoRentalCookie";

options.IdleTimeout = TimeSpan.FromMinutes(30);

});

builder.Services.AddControllersWithViews(options =>

{

options.CacheProfiles.Add("CacheProfile",

new CacheProfile()

{

Duration = 262

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseSession();

app.UseRouting();

app.UseAuthorization();

app.UseAuthentication();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}"

);

endpoints.MapControllerRoute(

name: "Disks",

pattern: "disks/{controller = Disks}/{action= Index}/{id?}"

);

}

);

app.Run();

**Листинг файла *Appsettings*.*json*:**

{

"ConnectionStrings": {

"SQLConnection": "Server=DESKTOP-I9HQNLA;Database=video\_rental;Trusted\_Connection=True;TrustServerCertificate=True"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Листинг файла *Clientele*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Clientele

{

public Clientele()

{

Takings = new HashSet<Taking>();

}

public int ClientId { get; set; }

public string Surname { get; set; } = null!;

public string Name { get; set; } = null!;

public string? Middlename { get; set; }

public string? Addres { get; set; }

public string? Phone { get; set; }

public string? Passport { get; set; }

public virtual ICollection<Taking> Takings { get; set; }

public override string ToString()

{

return ClientId + ", " + Surname + ", " + Name + ", " + Middlename + ", " + Addres + ", " + Phone + ", " + Passport;

}

}

}

**Листинг файла *Disk*.*cs*:**

using System.ComponentModel.DataAnnotations;

namespace VideoRentalWeb.DataModels

{

public partial class Disk

{

public Disk()

{

Pricelists = new HashSet<Pricelist>();

Takings = new HashSet<Taking>();

}

public int DiskId { get; set; }

public string Title { get; set; } = null!;

public string? CreationYear { get; set; }

public int Producer { get; set; }

public string MainActor { get; set; } = null!;

public DateTime Recording { get; set; }

public int GenreId { get; set; }

[Required]

public int DiskType { get; set; }

public virtual Type DiskTypeNavigation { get; set; } = null!;

public virtual Genre Genre { get; set; } = null!;

public virtual Producer ProducerNavigation { get; set; } = null!;

public virtual ICollection<Pricelist> Pricelists { get; set; }

public virtual ICollection<Taking> Takings { get; set; }

public override string ToString()

{

return DiskId + ", " + Title + ", " + CreationYear + ", " + Producer + ", " + MainActor + ", " + Recording.ToString() + ", " + GenreId + ", " + DiskType;

}

}

}

**Листинг файла *Genre*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Genre

{

public Genre()

{

Disks = new HashSet<Disk>();

}

public int GenreId { get; set; }

public string Title { get; set; } = null!;

public string? Description { get; set; }

public virtual ICollection<Disk> Disks { get; set; }

public override string ToString()

{

return GenreId + ", " + Title + ", " + Description;

}

}

}

**Листинг файла *Position*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Position

{

public Position()

{

staff = new HashSet<Staff>();

}

public int PositionId { get; set; }

public string Title { get; set; } = null!;

public virtual ICollection<Staff> staff { get; set; }

public override string ToString()

{

return PositionId + ", " + Title;

}

}

}

**Листинг файла *Pricelist*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Pricelist

{

public int PriceId { get; set; }

public int DiskId { get; set; }

public decimal? Price { get; set; }

public virtual Disk Disk { get; set; } = null!;

public override string ToString()

{

return PriceId + ", " + DiskId + ", " + Price;

}

}

}

**Листинг файла Program.cs:**

namespace VideoRentalWeb.DataModels

{

public partial class Producer

{

public Producer()

{

Disks = new HashSet<Disk>();

}

public int ProduceId { get; set; }

public string Manufacturer { get; set; } = null!;

public string? Country { get; set; }

public virtual ICollection<Disk> Disks { get; set; }

public override string ToString()

{

return ProduceId + ", " + Manufacturer + ", " + Country;

}

}

}

**Листинг файла *Staff*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Staff

{

public Staff()

{

Takings = new HashSet<Taking>();

}

public int StaffId { get; set; }

public string Surname { get; set; } = null!;

public string Name { get; set; } = null!;

public string? Middlename { get; set; }

public int? PositionId { get; set; }

public DateTime? DateOfEmployment { get; set; }

public virtual Position? Position { get; set; }

public virtual ICollection<Taking> Takings { get; set; }

public override string ToString()

{

return StaffId + ", " + Surname + ", " + Name + ", " + Middlename + ", " + PositionId + ", " +

DateOfEmployment.ToString();

}

}

}

**Листинг файла *Taking*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Taking

{

public int TakeId { get; set; }

public int ClientId { get; set; }

public int DiskId { get; set; }

public DateTime DateOfCapture { get; set; }

public DateTime ReturnDate { get; set; }

public bool PaymentMark { get; set; }

public bool RefundMark { get; set; }

public int StaffId { get; set; }

public virtual Clientele Client { get; set; } = null!;

public virtual Disk Disk { get; set; } = null!;

public virtual Staff Staff { get; set; } = null!;

public override string ToString()

{

return TakeId + ", " + ClientId + ", " + DiskId + ", " + DateOfCapture.ToString() + ", " +

ReturnDate.ToString() + ", " + PaymentMark.ToString() + ", " + RefundMark.ToString() + ", " +

StaffId.ToString();

}

}

}

**Листинг файла *Type*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class Type

{

public Type()

{

Disks = new HashSet<Disk>();

}

public int TypeId { get; set; }

public string Title { get; set; } = null!;

public string? Description { get; set; }

public virtual ICollection<Disk> Disks { get; set; }

public override string ToString()

{

return TypeId + ", " + Title + ", " + Description;

}

}

}

**Листинг файла *VideoRentalContext*.*cs*:**

using System.IO;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

namespace VideoRentalWeb.DataModels

{

public partial class VideoRentalContext : IdentityDbContext<User>

{

public VideoRentalContext()

{

}

public VideoRentalContext(DbContextOptions<VideoRentalContext> options) : base(options)

{

}

public virtual DbSet<Clientele> Clienteles { get; set; } = null!;

public virtual DbSet<Disk> Disks { get; set; } = null!;

public virtual DbSet<Genre> Genres { get; set; } = null!;

public virtual DbSet<Position> Positions { get; set; } = null!;

public virtual DbSet<Pricelist> Pricelists { get; set; } = null!;

public virtual DbSet<Producer> Producers { get; set; } = null!;

public virtual DbSet<Taking> Takings { get; set; } = null!;

public virtual DbSet<Type> Types { get; set; } = null!;

public virtual DbSet<ViewAllDisk> ViewAllDisks { get; set; } = null!;

public virtual DbSet<Staff> Staff { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

ConfigurationBuilder builder = new();

builder.SetBasePath(Directory.GetCurrentDirectory());

builder.AddJsonFile("D:\\Labs\\3 course\\5 sem\\РПБДИС\\Video\_rental\\VideoRentalWeb\\VideoRentalWeb\\appsettings.json");

IConfigurationRoot config = builder.Build();

string connectionString = config.GetConnectionString("SQLConnection");

\_ = optionsBuilder

.UseSqlServer(connectionString)

.Options;

optionsBuilder.LogTo(message => System.Diagnostics.Debug.WriteLine(message));

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Clientele>(entity =>

{

entity.HasKey(e => e.ClientId)

.HasName("PK\_\_Clientel\_\_E67E1A04315A897C");

entity.ToTable("Clientele");

entity.Property(e => e.ClientId).HasColumnName("ClientID");

entity.Property(e => e.Addres)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Middlename)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Name)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Passport)

.HasMaxLength(10)

.IsUnicode(false);

entity.Property(e => e.Phone)

.HasMaxLength(15)

.IsUnicode(false);

entity.Property(e => e.Surname)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Disk>(entity =>

{

entity.Property(e => e.DiskId).HasColumnName("DiskID");

entity.Property(e => e.CreationYear)

.HasMaxLength(4)

.IsUnicode(false);

entity.Property(e => e.GenreId).HasColumnName("GenreID");

entity.Property(e => e.MainActor)

.HasMaxLength(90)

.IsUnicode(false);

entity.Property(e => e.Recording).HasColumnType("date");

entity.Property(e => e.Title)

.HasMaxLength(30)

.IsUnicode(false);

entity.HasOne(d => d.DiskTypeNavigation)

.WithMany(p => p.Disks)

.HasForeignKey(d => d.DiskType)

.HasConstraintName("FK\_\_Disks\_\_DiskType\_\_3F466844");

entity.HasOne(d => d.Genre)

.WithMany(p => p.Disks)

.HasForeignKey(d => d.GenreId)

.HasConstraintName("FK\_\_Disks\_\_GenreID\_\_3E52440B");

entity.HasOne(d => d.ProducerNavigation)

.WithMany(p => p.Disks)

.HasForeignKey(d => d.Producer)

.HasConstraintName("FK\_\_Disks\_\_Producer\_\_3D5E1FD2");

});

modelBuilder.Entity<Genre>(entity =>

{

entity.Property(e => e.GenreId).HasColumnName("GenreID");

entity.Property(e => e.Description)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Title)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Position>(entity =>

{

entity.Property(e => e.PositionId).HasColumnName("PositionID");

entity.Property(e => e.Title)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Pricelist>(entity =>

{

entity.HasKey(e => e.PriceId)

.HasName("PK\_\_Pricelis\_\_4957584FC87A6E36");

entity.ToTable("Pricelist");

entity.Property(e => e.PriceId).HasColumnName("PriceID");

entity.Property(e => e.DiskId).HasColumnName("DiskID");

entity.Property(e => e.Price).HasColumnType("money");

entity.HasOne(d => d.Disk)

.WithMany(p => p.Pricelists)

.HasForeignKey(d => d.DiskId)

.HasConstraintName("FK\_\_Pricelist\_\_DiskI\_\_4316F928");

});

modelBuilder.Entity<Producer>(entity =>

{

entity.HasKey(e => e.ProduceId)

.HasName("PK\_\_Producer\_\_2EEBECB32A180669");

entity.Property(e => e.ProduceId).HasColumnName("ProduceID");

entity.Property(e => e.Country)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Manufacturer)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Taking>(entity =>

{

entity.HasKey(e => e.TakeId)

.HasName("PK\_\_Taking\_\_AC0C2240A69151F3");

entity.ToTable("Taking");

entity.Property(e => e.TakeId).HasColumnName("TakeID");

entity.Property(e => e.ClientId).HasColumnName("ClientID");

entity.Property(e => e.DateOfCapture).HasColumnType("date");

entity.Property(e => e.DiskId).HasColumnName("DiskID");

entity.Property(e => e.ReturnDate).HasColumnType("date");

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.HasOne(d => d.Client)

.WithMany(p => p.Takings)

.HasForeignKey(d => d.ClientId)

.HasConstraintName("FK\_\_Taking\_\_ClientID\_\_4D94879B");

entity.HasOne(d => d.Disk)

.WithMany(p => p.Takings)

.HasForeignKey(d => d.DiskId)

.HasConstraintName("FK\_\_Taking\_\_DiskID\_\_4E88ABD4");

entity.HasOne(d => d.Staff)

.WithMany(p => p.Takings)

.HasForeignKey(d => d.StaffId)

.HasConstraintName("FK\_\_Taking\_\_StaffID\_\_4F7CD00D");

});

modelBuilder.Entity<Type>(entity =>

{

entity.Property(e => e.TypeId).HasColumnName("TypeID");

entity.Property(e => e.Description)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Title)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<ViewAllDisk>(entity =>

{

entity.HasNoKey();

entity.ToView("View\_AllDisks");

entity.Property(e => e.Country)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.CreationYear)

.HasMaxLength(4)

.IsUnicode(false);

entity.Property(e => e.DiskId).HasColumnName("DiskID");

entity.Property(e => e.DiskTitle)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.GenreTitle)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.MainActor)

.HasMaxLength(90)

.IsUnicode(false);

entity.Property(e => e.Manufacturer)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Recording).HasColumnType("date");

entity.Property(e => e.TypeTitle)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Staff>(entity =>

{

entity.ToTable("Staff");

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.Property(e => e.DateOfEmployment).HasColumnType("date");

entity.Property(e => e.Middlename)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Name)

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.PositionId).HasColumnName("PositionID");

entity.Property(e => e.Surname)

.HasMaxLength(30)

.IsUnicode(false);

entity.HasOne(d => d.Position)

.WithMany(p => p.staff)

.HasForeignKey(d => d.PositionId)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Staff\_\_PositionI\_\_47DBAE45");

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

**Листинг файла *ViewAllDisk*.*cs*:**

namespace VideoRentalWeb.DataModels

{

public partial class ViewAllDisk

{

public int DiskId { get; set; }

public string DiskTitle { get; set; } = null!;

public string? CreationYear { get; set; }

public string Manufacturer { get; set; } = null!;

public string? Country { get; set; }

public string MainActor { get; set; } = null!;

public DateTime Recording { get; set; }

public string GenreTitle { get; set; } = null!;

public string TypeTitle { get; set; } = null!;

public override string ToString()

{

return DiskId + ", " + DiskTitle + ", " + CreationYear.ToString() + ", " + Manufacturer + ", " + Country + ", " + MainActor + ", " + Recording.ToString() + ", " + GenreTitle + ", " + TypeTitle;

}

}

}

**Листинг файла *AccountController*.*cs*:**

using System.Security.Claims;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Account;

namespace VideoRentalWeb.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

public AccountController(UserManager<User> userManager, SignInManager<User> signInManager)

{

this.\_userManager = userManager;

this.\_signInManager = signInManager;

}

[HttpGet]

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User

{

Email = model.Login,

UserName = model.Login

};

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

var claims = new List<Claim> { new Claim(ClaimTypes.Name, user.Email) };

// создаем объект ClaimsIdentity

ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, "Cookies");

// установка аутентификационных куки

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));

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

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpGet]

public IActionResult Login()

{

LoginViewModel model = new LoginViewModel();

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result = await \_signInManager.PasswordSignInAsync(model.Login, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

var claims = new List<Claim> { new Claim(ClaimTypes.Name, model.Login) };

// создаем объект ClaimsIdentity

ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, "Cookies");

// установка аутентификационных куки

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));

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

}

else

{

ModelState.AddModelError(string.Empty, "Inccorect login or password.");

}

}

return View(model);

}

[HttpGet]

public IActionResult Logout()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout(LogoutViewModel model)

{

await \_signInManager.SignOutAsync();

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

}

}

}

**Листинг файла *ClienteleController*.*cs*:**

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class ClienteleController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "clientele";

public ClienteleController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.ClientNameAsc, int page = 1)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new GenresFilterViewModel() { GenreTitle = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Genre).Name}-{page}-{sortState}-{filter.GenreTitle}";

if (!\_cache.TryGetValue(modelKey, out GenreViewModel model))

{

model = new GenreViewModel();

IQueryable<Genre> carMarks = GetSortedEntities(sortState, filter.GenreTitle);

int count = carMarks.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Genre>() : carMarks.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.GenresFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(GenresFilterViewModel filterModel, int page)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter != null)

{

filter.GenreTitle = filterModel.GenreTitle;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

GenreViewModel model = new GenreViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(GenreViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Genres.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre != null)

{

GenreViewModel model = new GenreViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = genre;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(GenreViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Genre genre = \_db.Genres.Find(model.Entity.GenreId);

if (genre != null)

{

genre.Title = model.Entity.Title;

\_db.Genres.Update(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Genres", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.DiskId == genre.GenreId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

GenreViewModel model = new GenreViewModel();

model.Entity = genre;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(GenreViewModel model)

{

Genre genre = await \_db.Genres.FindAsync(model.Entity.GenreId);

if (genre == null)

return NotFound();

\_db.Genres.Remove(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Genre genre)

{

bool firstFlag = true;

Genre tempgenre = \_db.Genres.FirstOrDefault(g => g.GenreId == genre.GenreId);

if (tempgenre != null)

{

if (tempgenre.GenreId != tempgenre.GenreId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Genre> GetSortedEntities(SortState sortState, string genreTitle)

{

IQueryable<Genre> genres = \_db.Genres.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

genres = genres.OrderBy(g => g.Title);

break;

case SortState.DiskTitleDesc:

genres = genres.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(genreTitle))

genres = genres.Where(g => g.Title.Contains(genreTitle)).AsQueryable();

return genres;

}

}

**Листинг файла *DisksController*.*cs*:**

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class DisksController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "disks";

public DisksController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.DiskTitleAsc, int page = 1)

{

DisksFilterViewModel filter = HttpContext.Session.Get<DisksFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new DisksFilterViewModel() { DiskTitle = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Disk).Name}-{page}-{sortState}-{filter.DiskTitle}";

if (!\_cache.TryGetValue(modelKey, out DiskViewModel model))

{

model = new DiskViewModel();

IQueryable<Disk> carMarks = GetSortedEntities(sortState, filter.DiskTitle);

int count = carMarks.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Disk>() : carMarks.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.DiskFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(DisksFilterViewModel filterModel, int page)

{

DisksFilterViewModel filter = HttpContext.Session.Get<DisksFilterViewModel>(FilterKey);

if (filter != null)

{

filter.DiskTitle = filterModel.DiskTitle;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

DiskViewModel model = new DiskViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(DiskViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

// if (ModelState.IsValid & CheckUniqueValues(model.Entity))

// {

await \_db.Disks.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

//}

// return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Disk disk = await \_db.Disks.FindAsync(id);

if (disk != null)

{

DiskViewModel model = new DiskViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = disk;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(DiskViewModel model)

{

Disk disk = \_db.Disks.Find(model.Entity.DiskId);

if (disk != null)

{

disk.Title = model.Entity.Title;

\_db.Disks.Update(disk);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Disks", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Disk disk = await \_db.Disks.FindAsync(id);

if (disk == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.DiskId == disk.DiskId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

DiskViewModel model = new DiskViewModel();

model.Entity = disk;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(DiskViewModel model)

{

Disk disk = await \_db.Disks.FindAsync(model.Entity.DiskId);

if (disk == null)

return NotFound();

\_db.Disks.Remove(disk);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Disk disk)

{

bool firstFlag = true;

Disk tempDisk = \_db.Disks.FirstOrDefault(g => g.DiskId == disk.DiskId);

if (tempDisk != null)

{

if (tempDisk.DiskId != disk.DiskId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Disk> GetSortedEntities(SortState sortState, string carMarkName)

{

IQueryable<Disk> carMarks = \_db.Disks.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

carMarks = carMarks.OrderBy(g => g.Title);

break;

case SortState.DiskTitleDesc:

carMarks = carMarks.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(carMarkName))

carMarks = carMarks.Where(g => g.Title.Contains(carMarkName)).AsQueryable();

return carMarks;

}

}

**Листинг файла *GenresController*.*cs*:**

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class GenresController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "genres";

public GenresController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.DiskTitleAsc, int page = 1)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new GenresFilterViewModel() { GenreTitle = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Genre).Name}-{page}-{sortState}-{filter.GenreTitle}";

if (!\_cache.TryGetValue(modelKey, out GenreViewModel model))

{

model = new GenreViewModel();

IQueryable<Genre> carMarks = GetSortedEntities(sortState, filter.GenreTitle);

int count = carMarks.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Genre>() : carMarks.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.GenresFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(GenresFilterViewModel filterModel, int page)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter != null)

{

filter.GenreTitle = filterModel.GenreTitle;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

GenreViewModel model = new GenreViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(GenreViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Genres.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre != null)

{

GenreViewModel model = new GenreViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = genre;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(GenreViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Genre genre = \_db.Genres.Find(model.Entity.GenreId);

if (genre != null)

{

genre.Title = model.Entity.Title;

\_db.Genres.Update(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Genres", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.DiskId == genre.GenreId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

GenreViewModel model = new GenreViewModel();

model.Entity = genre;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(GenreViewModel model)

{

Genre genre = await \_db.Genres.FindAsync(model.Entity.GenreId);

if (genre == null)

return NotFound();

\_db.Genres.Remove(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Genre genre)

{

bool firstFlag = true;

Genre tempgenre = \_db.Genres.FirstOrDefault(g => g.GenreId == genre.GenreId);

if (tempgenre != null)

{

if (tempgenre.GenreId != tempgenre.GenreId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Genre> GetSortedEntities(SortState sortState, string genreTitle)

{

IQueryable<Genre> genres = \_db.Genres.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

genres = genres.OrderBy(g => g.Title);

break;

case SortState.DiskTitleDesc:

genres = genres.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(genreTitle))

genres = genres.Where(g => g.Title.Contains(genreTitle)).AsQueryable();

return genres;

}

}

**Листинг файла *HomeController*.*cs*:**

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.Models;

namespace VideoRentalWeb.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**Листинг файла *PositionsController*.*cs*:**

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class PositionsController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "positions";

public PositionsController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.GenreTitleAsc, int page = 1)

{

PositionsFilterViewModel filter = HttpContext.Session.Get<PositionsFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new PositionsFilterViewModel() { Title = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Position).Name}-{page}-{sortState}-{filter.Title}";

if (!\_cache.TryGetValue(modelKey, out PositionsViewModel model))

{

model = new PositionsViewModel();

IQueryable<Position> positions = GetSortedEntities(sortState, filter.Title);

int count = positions.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Position>() : positions.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.PositionsFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(PositionsFilterViewModel filterModel, int page)

{

PositionsFilterViewModel filter = HttpContext.Session.Get<PositionsFilterViewModel>(FilterKey);

if (filter != null)

{

filter.Title = filterModel.Title;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

PositionsViewModel model = new PositionsViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(PositionsViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Genres.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Position genre = await \_db.Positions.FindAsync(id);

if (genre != null)

{

PositionsViewModel model = new PositionsViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = genre;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(PositionsViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Position position = \_db.Positions.Find(model.Entity.PositionId);

if (position != null)

{

position.Title = model.Entity.Title;

\_db.Positions.Update(position);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Positions", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Position position = await \_db.Positions.FindAsync(id);

if (position == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

PositionsViewModel model = new PositionsViewModel();

model.Entity = position;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(PositionsViewModel model)

{

Position genre = await \_db.Positions.FindAsync(model.Entity.PositionId);

if (genre == null)

return NotFound();

\_db.Positions.Remove(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Position position)

{

bool firstFlag = true;

Position tempgenre = \_db.Positions.FirstOrDefault(g => g.PositionId == position.PositionId);

if (tempgenre != null)

{

if (tempgenre.PositionId != position.PositionId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Position> GetSortedEntities(SortState sortState, string title)

{

IQueryable<Position> positions = \_db.Positions.AsQueryable();

switch (sortState)

{

case SortState.PositionTitleAsc:

positions = positions.OrderBy(g => g.Title);

break;

case SortState.PositionDesc:

positions = positions.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(title))

positions = positions.Where(g => g.Title.Contains(title)).AsQueryable();

return positions;

}

}

**Листинг файла *PricelistController*.*cs*:**

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class PricelistController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "pricelist";

public PricelistController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.PricelistDiscIdAsc, int page = 1)

{

PricelistFilterViewModel filter = HttpContext.Session.Get<PricelistFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new PricelistFilterViewModel() { Price = 0 };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Pricelist).Name}-{page}-{sortState}-{filter.Price}";

if (!\_cache.TryGetValue(modelKey, out PricelistViewModel model))

{

model = new PricelistViewModel();

IQueryable<Pricelist> pricelists = GetSortedEntities(sortState, filter.Price);

int count = pricelists.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Pricelist>() : pricelists.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.PricelistFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(PricelistFilterViewModel filterModel, int page)

{

PricelistFilterViewModel filter = HttpContext.Session.Get<PricelistFilterViewModel>(FilterKey);

if (filter != null)

{

filter.Price = filterModel.Price;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

PricelistViewModel model = new PricelistViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(PricelistViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Pricelists.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Pricelist pricelist = await \_db.Pricelists.FindAsync(id);

if (pricelist != null)

{

PricelistViewModel model = new PricelistViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = pricelist;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(PricelistViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Pricelist pricelist = \_db.Pricelists.Find(model.Entity.PriceId);

if (pricelist != null)

{

pricelist.Price = model.Entity.Price;

\_db.Pricelists.Update(pricelist);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Pricelist", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Pricelist pricelist = await \_db.Pricelists.FindAsync(id);

if (pricelist == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

PricelistViewModel model = new PricelistViewModel();

model.Entity = pricelist;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(PricelistViewModel model)

{

Pricelist pricelist = await \_db.Pricelists.FindAsync(model.Entity.PriceId);

if (pricelist == null)

return NotFound();

\_db.Pricelists.Remove(pricelist);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Pricelist pricelist)

{

bool firstFlag = true;

Pricelist tempgenre = \_db.Pricelists.FirstOrDefault(g => g.DiskId == pricelist.DiskId);

if (tempgenre != null)

{

if (pricelist.PriceId != tempgenre.PriceId)

{

ModelState.AddModelError(string.Empty, "Another entity have this disk Id. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Pricelist> GetSortedEntities(SortState sortState, string price)

{

IQueryable<Pricelist> pricelists = \_db.Pricelists.AsQueryable();

switch (sortState)

{

case SortState.PricelistDiscIdAsc:

pricelists = pricelists.OrderBy(g => g.DiskId);

break;

case SortState.PricelistDiscIdDesc:

pricelists = pricelists.OrderByDescending(g => g.DiskId);

break;

}

if (!string.IsNullOrEmpty(price))

pricelists = pricelists.Where(g => g.Price.Equals(price)).AsQueryable();

return pricelists;

}

}

**Листинг файла *ProducersController*.*cs*:**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class ProducersController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "producers";

public ProducersController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.ManufacturerAsc, int page = 1)

{

ProducersFilterViewModel filter = HttpContext.Session.Get<ProducersFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new ProducersFilterViewModel() { Manufacturer = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Producer).Name}-{page}-{sortState}-{filter.Manufacturer}";

if (!\_cache.TryGetValue(modelKey, out ProducersViewModel model))

{

model = new ProducersViewModel();

IQueryable<Producer> producers = GetSortedEntities(sortState, filter.Manufacturer);

int count = producers.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Producer>() : producers.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.ProducersFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(ProducersFilterViewModel filterModel, int page)

{

ProducersFilterViewModel filter = HttpContext.Session.Get<ProducersFilterViewModel>(FilterKey);

if (filter != null)

{

filter.Manufacturer = filterModel.Manufacturer;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

ProducersViewModel model = new ProducersViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(ProducersViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Producers.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Producer producer = await \_db.Producers.FindAsync(id);

if (producer != null)

{

ProducersViewModel model = new ProducersViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = producer;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(ProducersViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Producer producer = \_db.Producers.Find(model.Entity.ProduceId);

if (producer != null)

{

producer.Manufacturer = model.Entity.Manufacturer;

producer.Country = model.Entity.Country;

\_db.Producers.Update(producer);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Producers", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Producer producer = await \_db.Producers.FindAsync(id);

if (producer == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.Producer == producer.ProduceId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

ProducersViewModel model = new ProducersViewModel();

model.Entity = producer;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(ProducersViewModel model)

{

Producer producer = await \_db.Producers.FindAsync(model.Entity.ProduceId);

if (producer == null)

return NotFound();

\_db.Producers.Remove(producer);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Producer producer)

{

bool firstFlag = true;

Producer tempprod = \_db.Producers.FirstOrDefault(g => g.Manufacturer == producer.Manufacturer);

if (tempprod != null)

{

if (tempprod.ProduceId != producer.ProduceId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Producer> GetSortedEntities(SortState sortState, string genreTitle)

{

IQueryable<Producer> producers = \_db.Producers.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

producers = producers.OrderBy(g => g.Manufacturer);

break;

case SortState.DiskTitleDesc:

producers = producers.OrderByDescending(g => g.Manufacturer);

break;

}

if (!string.IsNullOrEmpty(genreTitle))

producers = producers.Where(g => g.Manufacturer.Contains(genreTitle)).AsQueryable();

return producers;

}

}

**Листинг файла *RolesController*.*cs*:**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models;

namespace VideoRentalWeb.Controllers

{

public class RolesController : Controller

{

private readonly RoleManager<IdentityRole> \_roleManager;

private readonly UserManager<User> \_userManager;

public RolesController(RoleManager<IdentityRole> roleManager, UserManager<User> userManager)

{

this.\_roleManager = roleManager;

this.\_userManager = userManager;

}

public IActionResult Index()

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

RoleViewModel model = new RoleViewModel

{

Roles = \_roleManager.Roles.ToList()

};

return View(model);

}

else

{

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

}

}

public IActionResult Create()

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

RoleViewModel model = new RoleViewModel();

return View(model);

}

else

{

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

}

}

[HttpPost]

public async Task<IActionResult> Create(RoleViewModel model)

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

if (!string.IsNullOrEmpty(model.RoleName))

{

var result = await \_roleManager.CreateAsync(new IdentityRole(model.RoleName));

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

ModelState.AddModelError(string.Empty, error.Description);

}

}

return View(model);

}

else

{

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

}

}

[HttpPost]

public async Task<IActionResult> Delete(string name)

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

IdentityRole role = await \_roleManager.FindByNameAsync(name);

if (role != null)

{

var result = await \_roleManager.DeleteAsync(role);

}

return RedirectToAction("Index");

}

else

{

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

}

}

public IActionResult Users()

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

return View(\_userManager.Users.ToList());

}

else

{

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

}

}

public async Task<IActionResult> Edit(string userId)

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

var userRoles = await \_userManager.GetRolesAsync(user);

var roles = \_roleManager.Roles.ToList();

RoleViewModel model = new RoleViewModel

{

Roles = roles,

UserRoles = userRoles,

User = user

};

return View(model);

}

return NotFound();

}

else

{

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

}

}

[HttpPost]

public async Task<IActionResult> Edit(string userId, List<string> roles)

{

var currentUser = \_userManager.GetUserAsync(User).Result;

// Проверка наличия роли Admin у текущего пользователя

if (\_userManager.IsInRoleAsync(currentUser, "Admin").Result)

{

User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

var userRoles = await \_userManager.GetRolesAsync(user);

var allRoles = \_roleManager.Roles.ToList();

var addedRoles = roles.Except(userRoles);

var removedRoles = userRoles.Except(roles);

await \_userManager.AddToRolesAsync(user, addedRoles);

await \_userManager.RemoveFromRolesAsync(user, removedRoles);

return RedirectToAction("Users");

}

return NotFound();

}

else

{

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

}

}

}

}

**Листинг файла *StaffController*.*cs*:**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class StaffController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "genres";

public StaffController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.StaffNameAsc, int page = 1)

{

StaffFilterViewModel filter = HttpContext.Session.Get<StaffFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new StaffFilterViewModel() { Name = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Staff).Name}-{page}-{sortState}-{filter.Name}";

if (!\_cache.TryGetValue(modelKey, out StaffViewModel model))

{

model = new StaffViewModel();

IQueryable<Staff> carMarks = GetSortedEntities(sortState, filter.Name);

int count = carMarks.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Staff>() : carMarks.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.StaffFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(StaffFilterViewModel filterModel, int page)

{

StaffFilterViewModel filter = HttpContext.Session.Get<StaffFilterViewModel>(FilterKey);

if (filter != null)

{

filter.Name = filterModel.Name;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

StaffViewModel model = new StaffViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(StaffViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Staff.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Staff staff = await \_db.Staff.FindAsync(id);

if (staff != null)

{

StaffViewModel model = new StaffViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = staff;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(StaffViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Staff staff = \_db.Staff.Find(model.Entity.StaffId);

if (staff != null)

{

staff.Name = model.Entity.Name;

\_db.Staff.Update(staff);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Staff", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.DiskId == genre.GenreId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

GenreViewModel model = new GenreViewModel();

model.Entity = genre;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(GenreViewModel model)

{

Genre genre = await \_db.Genres.FindAsync(model.Entity.GenreId);

if (genre == null)

return NotFound();

\_db.Genres.Remove(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Staff genre)

{

bool firstFlag = true;

Genre tempgenre = \_db.Genres.FirstOrDefault(g => g.GenreId == genre.GenreId);

if (tempgenre != null)

{

if (tempgenre.GenreId != tempgenre.GenreId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Staff> GetSortedEntities(SortState sortState, string genreTitle)

{

IQueryable<Genre> genres = \_db.Genres.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

genres = genres.OrderBy(g => g.Title);

break;

case SortState.DiskTitleDesc:

genres = genres.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(genreTitle))

genres = genres.Where(g => g.Title.Contains(genreTitle)).AsQueryable();

return genres;

}

}

**Листинг файла *TakingsController*.*cs*:**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

namespace VideoRentalWeb.Controllers;

public class TakingsController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "genres";

public TakingsController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.DiskTitleAsc, int page = 1)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new GenresFilterViewModel() { GenreTitle = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Genre).Name}-{page}-{sortState}-{filter.GenreTitle}";

if (!\_cache.TryGetValue(modelKey, out GenreViewModel model))

{

model = new GenreViewModel();

IQueryable<Genre> carMarks = GetSortedEntities(sortState, filter.GenreTitle);

int count = carMarks.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Genre>() : carMarks.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.GenresFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(GenresFilterViewModel filterModel, int page)

{

GenresFilterViewModel filter = HttpContext.Session.Get<GenresFilterViewModel>(FilterKey);

if (filter != null)

{

filter.GenreTitle = filterModel.GenreTitle;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

GenreViewModel model = new GenreViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(GenreViewModel model)

{

foreach (var entry in ModelState)

{

var key = entry.Key; // Название свойства

var errors = entry.Value.Errors.Select(e => e.ErrorMessage).ToList(); // Список ошибок для свойства

// Далее можно использовать key и errors в соответствии с вашими потребностями

Console.WriteLine($"Property: {key}, Errors: {string.Join(", ", errors)}");

}

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Genres.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre != null)

{

GenreViewModel model = new GenreViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = genre;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(GenreViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Genre genre = \_db.Genres.Find(model.Entity.GenreId);

if (genre != null)

{

genre.Title = model.Entity.Title;

\_db.Genres.Update(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Genres", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Genre genre = await \_db.Genres.FindAsync(id);

if (genre == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Disks.Any(s => s.DiskId == genre.GenreId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

GenreViewModel model = new GenreViewModel();

model.Entity = genre;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(GenreViewModel model)

{

Genre genre = await \_db.Genres.FindAsync(model.Entity.GenreId);

if (genre == null)

return NotFound();

\_db.Genres.Remove(genre);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Genre genre)

{

bool firstFlag = true;

Genre tempgenre = \_db.Genres.FirstOrDefault(g => g.GenreId == genre.GenreId);

if (tempgenre != null)

{

if (tempgenre.GenreId != tempgenre.GenreId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Genre> GetSortedEntities(SortState sortState, string genreTitle)

{

IQueryable<Genre> genres = \_db.Genres.AsQueryable();

switch (sortState)

{

case SortState.DiskTitleAsc:

genres = genres.OrderBy(g => g.Title);

break;

case SortState.DiskTitleDesc:

genres = genres.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(genreTitle))

genres = genres.Where(g => g.Title.Contains(genreTitle)).AsQueryable();

return genres;

}

}

**Листинг файла *TypesController*.*cs*:**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Infrastructure;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

using VideoRentalWeb.Models.Filters;

using VideoRentalWeb.Services;

using Type = VideoRentalWeb.DataModels.Type;

namespace VideoRentalWeb.Controllers;

public class TypesController : Controller

{

private readonly VideoRentalContext \_db;

private readonly CacheProvider \_cache;

private const string FilterKey = "types";

public TypesController(VideoRentalContext context, CacheProvider cacheProvider)

{

\_db = context;

\_cache = cacheProvider;

}

public IActionResult Index(SortState sortState = SortState.TypeTitleAsc, int page = 1)

{

TypeFilterViewModel filter = HttpContext.Session.Get<TypeFilterViewModel>(FilterKey);

if (filter == null)

{

filter = new TypeFilterViewModel() { TypeTitle = string.Empty };

HttpContext.Session.Set(FilterKey, filter);

}

string modelKey = $"{typeof(Type).Name}-{page}-{sortState}-{filter.TypeTitle}";

if (!\_cache.TryGetValue(modelKey, out TypeViewModel model))

{

model = new TypeViewModel();

IQueryable<Type> types = GetSortedEntities(sortState, filter.TypeTitle);

int count = types.Count();

int pageSize = 10;

model.PageViewModel = new PageViewModel(page, count, pageSize);

model.Entities = count == 0 ? new List<Type>() : types.Skip((model.PageViewModel.CurrentPage - 1) \* pageSize).Take(pageSize).ToList();

model.SortViewModel = new SortViewModel(sortState);

model.TypeFilterViewModel = filter;

\_cache.Set(modelKey, model);

}

return View(model);

}

[HttpPost]

public IActionResult Index(TypeFilterViewModel filterModel, int page)

{

TypeFilterViewModel filter = HttpContext.Session.Get<TypeFilterViewModel>(FilterKey);

if (filter != null)

{

filter.TypeTitle = filterModel.TypeTitle;

HttpContext.Session.Remove(FilterKey);

HttpContext.Session.Set(FilterKey, filter);

}

return RedirectToAction("Index", new { page });

}

public IActionResult Create(int page)

{

TypeViewModel model = new TypeViewModel()

{

PageViewModel = new PageViewModel { CurrentPage = page }

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(TypeViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

await \_db.Types.AddAsync(model.Entity);

await \_db.SaveChangesAsync();

\_cache.Clean();

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

}

return View(model);

}

public async Task<IActionResult> Edit(int id, int page)

{

Type type = await \_db.Types.FindAsync(id);

if (type != null)

{

TypeViewModel model = new TypeViewModel();

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.Entity = type;

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(TypeViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

Type type = \_db.Types.Find(model.Entity.TypeId);

if (type != null)

{

type.Title = model.Entity.Title;

\_db.Types.Update(type);

await \_db.SaveChangesAsync();

\_cache.Clean();

return RedirectToAction("Index", "Types", new { page = model.PageViewModel.CurrentPage });

}

else

{

return NotFound();

}

}

return View(model);

}

public async Task<IActionResult> Delete(int id, int page)

{

Type type = await \_db.Types.FindAsync(id);

if (type == null)

return NotFound();

bool deleteFlag = false;

string message = "Do you want to delete this entity";

if (\_db.Types.Any(s => s.TypeId == type.TypeId))

message = "This entity has entities, which dependents from this. Do you want to delete this entity and other, which dependents from this?";

TypeViewModel model = new TypeViewModel();

model.Entity = type;

model.PageViewModel = new PageViewModel { CurrentPage = page };

model.DeleteViewModel = new DeleteViewModel { Message = message, IsDeleted = deleteFlag };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(TypeViewModel model)

{

Type type = await \_db.Types.FindAsync(model.Entity.TypeId);

if (type == null)

return NotFound();

\_db.Types.Remove(type);

await \_db.SaveChangesAsync();

\_cache.Clean();

model.DeleteViewModel = new DeleteViewModel { Message = "The entity was successfully deleted.", IsDeleted = true };

return View(model);

}

private bool CheckUniqueValues(Type type)

{

bool firstFlag = true;

Type tempType = \_db.Types.FirstOrDefault(g => g.Title == type.Title);

if (tempType != null)

{

if (tempType.TypeId != type.TypeId)

{

ModelState.AddModelError(string.Empty, "Another entity have this name. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

private IQueryable<Type> GetSortedEntities(SortState sortState, string typeName)

{

IQueryable<Type> types = \_db.Types.AsQueryable();

switch (sortState)

{

case SortState.TypeTitleAsc:

types = types.OrderBy(g => g.Title);

break;

case SortState.TypeTitleDesc:

types = types.OrderByDescending(g => g.Title);

break;

}

if (!string.IsNullOrEmpty(typeName))

types = types.Where(g => g.Title.Contains(typeName)).AsQueryable();

return types;

}

}

**Листинг файла *UsersController*.*cs*:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models;

using VideoRentalWeb.Models.Entities;

namespace VideoRentalWeb.Controllers

{

[Authorize(Roles = "admin")]

public class UsersController : Controller

{

private readonly UserManager<User> manager;

private readonly SignInManager<User> inManager;

public UsersController(UserManager<User> userManager, SignInManager<User> signInManager)

{

manager = userManager;

inManager = signInManager;

}

public IActionResult Index()

{

IEnumerable<User> users = manager.Users.ToList();

UsersViewModel model = new UsersViewModel

{

Entities = users

};

return View(model);

}

public IActionResult Create()

{

UsersViewModel model = new UsersViewModel

{

Entity = new User()

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Create(UsersViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

model.Entity.UserName = model.Entity.Email;

var result = await manager.CreateAsync(model.Entity, model.Password);

if (result.Succeeded)

{

User user = await manager.FindByNameAsync(model.Entity.Email);

if (user != null)

{

await manager.AddToRoleAsync(user, "user");

}

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

ModelState.AddModelError(string.Empty, error.Description);

}

}

return View(model);

}

public async Task<IActionResult> Edit(string id)

{

User user = await manager.FindByIdAsync(id);

if (user == null)

return NotFound();

UsersViewModel model = new UsersViewModel

{

Entity = user

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Edit(UsersViewModel model)

{

if (ModelState.IsValid & CheckUniqueValues(model.Entity))

{

User user = await manager.FindByIdAsync(model.Entity.Id.ToString());

if (user != null)

{

user.Email = model.Entity.Email;

user.UserName = model.Entity.Email;

var result = await manager.UpdateAsync(user);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

}

return View(model);

}

public async Task<IActionResult> Delete(string id)

{

User user = await manager.FindByIdAsync(id);

if (user == null)

return NotFound();

UsersViewModel model = new UsersViewModel

{

Entity = user,

DeleteViewModel = new DeleteViewModel

{

Message = "Do you want to delete this user?",

IsDeleted = true

}

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Delete(UsersViewModel model)

{

User user = await manager.FindByIdAsync(model.Entity.Id.ToString());

if (user == null)

return NotFound();

var result = await manager.DeleteAsync(user);

if (result.Succeeded)

{

model.DeleteViewModel.IsDeleted = false;

if (User.Identity.IsAuthenticated & User.Identity.Name == user.UserName)

{

await inManager.SignOutAsync();

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

}

return View(model);

}

else

{

foreach (var error in result.Errors)

ModelState.AddModelError(string.Empty, error.Description);

}

model.DeleteViewModel.IsDeleted = true;

return View(model);

}

public async Task<IActionResult> ChangePassword(string id)

{

User user = await manager.FindByIdAsync(id);

if (user == null)

return NotFound();

UsersViewModel model = new UsersViewModel

{

Entity = user

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> ChangePassword(UsersViewModel model)

{

if (ModelState.IsValid)

{

User user = await manager.FindByIdAsync(model.Entity.Id.ToString());

if (user == null)

return NotFound();

var result = await manager.ChangePasswordAsync(user, model.Password, model.NewPassword);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

ModelState.AddModelError(string.Empty, error.Description);

}

}

return View(model);

}

private bool CheckUniqueValues(User user)

{

bool firstFlag = true;

IEnumerable<User> users = manager.Users.ToList();

User tempUser = users.FirstOrDefault(u => u.Email == user.Email);

if (tempUser != null)

{

if (tempUser.Id != user.Id)

{

ModelState.AddModelError(string.Empty, "Another entity have this email. Please replace this to another.");

firstFlag = false;

}

}

if (firstFlag)

return true;

else

return false;

}

}

}

**Листинг файла *SessionExtensions*.*cs*:**

using System.Text.Json;

namespace VideoRentalWeb.Infrastructure

{

public static class SessionExtensions

{

public static void Set<T>(this ISession session, string key, T value)

{

session.SetString(key, JsonSerializer.Serialize(value));

}

public static T Get<T>(this ISession session, string key)

{

var value = session.GetString(key);

return value == null ? default : JsonSerializer.Deserialize<T>(value);

}

}

}

**Листинг файла *LoginViewModel*.*cs*:**

using System.ComponentModel.DataAnnotations;

namespace VideoRentalWeb.Models.Account

{

public class LoginViewModel

{

[Required]

[Display(Name = "Login")]

[DataType(DataType.EmailAddress)]

public string Login { get; set; }

[Required]

[Display(Name = "Password")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Remember?")]

public bool RememberMe { get; set; }

}

}

**Листинг файла *LogoutViewModel*.*cs*:**

namespace VideoRentalWeb.Models.Account

{

public class LogoutViewModel

{

}

}

**Листинг файла *RegisterViewModel*.*cs*:**

using System.ComponentModel.DataAnnotations;

namespace VideoRentalWeb.Models.Account

{

public class RegisterViewModel

{

[Required]

[Display(Name = "Login")]

[DataType(DataType.EmailAddress)]

public string Login { get; set; }

[Required]

[Display(Name = "Password")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required]

[Display(Name = "Confirm password")]

[DataType(DataType.Password)]

public string PasswordConfirm { get; set; }

}

}

**Листинг файла *ClienteleViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class ClienteleViewModel : IEntitiesViewModel<Clientele>

{

[Display(Name = "Clientele")]

public IEnumerable<Clientele>? Entities { get; set; }

[Display(Name = "Clientele")]

public Clientele Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public ClienteleFilterViewModel? ClienteleFilterViewModel { get; set; }

}

}

**Листинг файла *DiskViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

using Type = VideoRentalWeb.DataModels.Type;

namespace VideoRentalWeb.Models.Entities

{

public class DiskViewModel : IEntitiesViewModel<Disk>

{

[Display(Name = "Disks")]

public IEnumerable<Disk>? Entities { get; set; }

[Display(Name = "Disk")]

public Disk Entity { get; set; }

[Display(Name = "Genres")]

public IEnumerable<Genre>? GenreList { get; set; }

[Display(Name = "Types")]

public IEnumerable<Type>? TypesList { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public DisksFilterViewModel? DiskFilterViewModel { get; set; }

}

}

**Листинг файла *GenreViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class GenreViewModel : IEntitiesViewModel<Genre>

{

[Display(Name = "Genres")]

public IEnumerable<Genre>? Entities { get; set; }

[Display(Name = "Genre")]

public Genre Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public GenresFilterViewModel? GenresFilterViewModel { get; set; }

}

}

**Листинг файла *IEntitiesViewModel*.*cs*:**

using System.Collections.Generic;

using VideoRentalWeb.DataModels;

namespace VideoRentalWeb.Models.Entities

{

public interface IEntitiesViewModel<T>

{

IEnumerable<T> Entities { get; set; }

T Entity { get; set; }

PageViewModel PageViewModel { get; set; }

DeleteViewModel DeleteViewModel { get; set; }

SortViewModel SortViewModel { get; set; }

}

}

**Листинг файла *PositionsViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class PositionsViewModel : IEntitiesViewModel<Position>

{

[Display(Name = "Positions")]

public IEnumerable<Position>? Entities { get; set; }

[Display(Name = "Position")]

public Position Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public PositionsFilterViewModel? PositionsFilterViewModel{ get; set; }

}

}

**Листинг файла *PricelistViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class PricelistViewModel : IEntitiesViewModel<Pricelist>

{

[Display(Name = "Pricelist")]

public IEnumerable<Pricelist>? Entities { get; set; }

[Display(Name = "Price")]

public Pricelist Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public PricelistFilterViewModel? PricelistFilterViewModel { get; set; }

}

}

**Листинг файла *ProducersViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class ProducersViewModel : IEntitiesViewModel<Producer>

{

[Display(Name = "Producers")]

public IEnumerable<Producer>? Entities { get; set; }

[Display(Name = "Producer")]

public Producer Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public ProducersFilterViewModel? ProducersFilterViewModel { get; set; }

}

}

**Листинг файла *StaffViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class StaffViewModel : IEntitiesViewModel<Staff>

{

[Display(Name = "Staff")]

public IEnumerable<Staff>? Entities { get; set; }

[Display(Name = "Staff")]

public Staff Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public StaffFilterViewModel? StaffFilterViewModel { get; set; }

}

}

**Листинг файла *TakingViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

using VideoRentalWeb.Models.Filters;

namespace VideoRentalWeb.Models.Entities

{

public class TakingViewModel : IEntitiesViewModel<Taking>

{

[Display(Name = "Takings")]

public IEnumerable<Taking>? Entities { get; set; }

[Display(Name = "Taking")]

public Taking Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel? DeleteViewModel { get; set; }

public SortViewModel? SortViewModel { get; set; }

public DisksFilterViewModel? DiskFilterViewModel { get; set; }

}

}

**Листинг файла *TypeViewModel*.*cs*:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.Models.Filters;

using Type = VideoRentalWeb.DataModels.Type;

namespace VideoRentalWeb.Models.Entities

{

public class TypeViewModel : IEntitiesViewModel<Type>

{

[Display(Name = "Types")]

public IEnumerable<Type> Entities { get; set; }

[Display(Name = "Type")]

public Type Entity { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel DeleteViewModel { get; set; }

public SortViewModel SortViewModel { get; set; }

public TypeFilterViewModel TypeFilterViewModel { get; set; }

}

}

**Листинг файла *UsersViewModel*.*cs*:**

using System.ComponentModel.DataAnnotations;

using VideoRentalWeb.DataModels;

namespace VideoRentalWeb.Models.Entities

{

public class UsersViewModel : IEntitiesViewModel<User>

{

[Display(Name = "Users")]

public IEnumerable<User> Entities { get; set; }

[Display(Name = "User")]

public User Entity { get; set; }

[Required]

[Display(Name = "Password")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required]

[Display(Name = "New password")]

[DataType(DataType.Password)]

public string NewPassword { get; set; }

public PageViewModel PageViewModel { get; set; }

public DeleteViewModel DeleteViewModel { get; set; }

public SortViewModel SortViewModel { get; set; }

}

}

**Листинг файла *DeleteViewModel*.*cs*:**

namespace VideoRentalWeb.Models

{

public class DeleteViewModel

{

public string Message { get; set; }

public bool IsDeleted { get; set; }

}

}

**Листинг файла *ErrorViewModel*.*cs*:**

namespace VideoRentalWeb.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

**Листинг файла *PageViewModel*.*cs*:**

namespace VideoRentalWeb.Models

{

public class PageViewModel

{

public int CurrentPage { get; set; }

public int TotalPages { get; private set; }

public PageViewModel()

{ }

public PageViewModel(int page, int count, int pageSize = 10)

{

CurrentPage = page;

TotalPages = (int)Math.Ceiling(count / (double)pageSize);

if (CurrentPage <= 0)

CurrentPage = 1;

if (CurrentPage > TotalPages)

CurrentPage = TotalPages;

}

public bool HasPreviousPage

{

get

{

return (CurrentPage > 1);

}

}

public bool HasNextPage

{

get

{

return (CurrentPage < TotalPages);

}

}

}

}

**Листинг файла *RoleViewModel*.*cs*:**

using System.ComponentModel.DataAnnotations;

using Microsoft.AspNetCore.Identity;

using VideoRentalWeb.DataModels;

namespace VideoRentalWeb.Models

{

public class RoleViewModel

{

public User User { get; set; }

[Display(Name = "Roles")]

public IEnumerable<IdentityRole> Roles { get; set; }

[Display(Name = "User roles")]

public IList<string> UserRoles { get; set; }

[Required]

[Display(Name = "Role")]

[StringLength(32, MinimumLength = 4, ErrorMessage = "The role name length must be more 4")]

public string RoleName { get; set; }

[Display(Name = "Modification")]

public string Modification { get; set; }

}

}

**Листинг файла *SortViewModel*.*cs*:**

namespace VideoRentalWeb.Models

{

public enum SortState

{

No,

//Genres

GenreTitleAsc,

GenreTitleDesc,

//Disks

DiskTitleAsc,

DiskCreationYearAsc,

DiskProducerAsc,

MainActorAsc,

RecordingDateAsc,

GenreIdAsc,

TypeIdAsc,

DiskTitleDesc,

DiskCreationYearDesc,

DiskProducerDesc,

MainActorDesc,

RecordingDateDesc,

GenreIdDesc,

TypeIdDesc,

//Clientele

ClientSurnameAsc,

ClientNameAsc,

ClientMiddlenameAsc,

ClientAddressAsc,

ClientPhoneAsc,

ClientPassportAsc,

ClientPhoneDesc,

ClientPassportDesc,

ClientAddressDesc,

ClientMiddlenameDesc,

ClientNameDesc,

ClientSurnameDesc,

//Types

TypeTitleAsc,

TypeTitleDesc,

//Positions

PositionTitleAsc,

PositionDesc,

//Staff

StaffNameAsc,

StaffNameDesc,

StaffSurnameAsc,

StaffSurnameDesc,

//Pricelist

PricelistDiscIdAsc,

PricelistDiscIdDesc,

//Producers

ManufacturerAsc,

ManufacturerDesc,

}

public class SortViewModel

{

//Genre

public SortState GenreTitleSort { get; set; }

//Disks

public SortState DiskTitleSort { get; set; }

public SortState DiskCreationYearSort { get; set; }

public SortState DiskProducerSort { get; set; }

public SortState MainActorSort { get; set; }

public SortState RecordingDateSort { get; set; }

public SortState DiskGenreIdSort { get; set; }

public SortState DiskTypeIdSort { get; set; }

//Types

public SortState TypeTitleSort { get; set; }

public SortState CurrentState { get; set; }

public SortViewModel(SortState state)

{

//Genre

GenreTitleSort = state == SortState.GenreTitleAsc ? SortState.GenreTitleDesc : SortState.GenreTitleAsc;

CurrentState = state;

//Disks

DiskTitleSort = state == SortState.DiskTitleAsc ? SortState.DiskTitleDesc : SortState.DiskTitleAsc;

CurrentState = state;

DiskCreationYearSort = state == SortState.DiskCreationYearAsc ? SortState.DiskCreationYearDesc : SortState.DiskCreationYearAsc;

CurrentState = state;

DiskProducerSort = state == SortState.DiskProducerAsc ? SortState.DiskProducerDesc : SortState.DiskProducerAsc;

CurrentState = state;

MainActorSort = state == SortState.MainActorAsc ? SortState.MainActorDesc : SortState.MainActorAsc;

CurrentState = state;

RecordingDateSort = state == SortState.RecordingDateAsc ? SortState.RecordingDateDesc : SortState.RecordingDateAsc;

CurrentState = state;

DiskGenreIdSort = state == SortState.GenreIdAsc ? SortState.GenreIdDesc : SortState.GenreIdAsc;

CurrentState = state;

DiskTypeIdSort = state == SortState.TypeIdAsc ? SortState.TypeIdDesc : SortState.TypeIdAsc;

CurrentState = state;

//Types

TypeTitleSort = state == SortState.TypeTitleAsc ? SortState.TypeTitleDesc : SortState.TypeTitleAsc;

CurrentState = state;

}

}

}

**Листинг файла *CacheProvider*.*cs*:**

using Microsoft.Extensions.Caching.Memory;

using Microsoft.Extensions.Primitives;

namespace VideoRentalWeb.Services

{

public class CacheProvider

{

private static CancellationTokenSource \_resetCacheToken = new CancellationTokenSource();

private readonly IMemoryCache \_cache;

public CacheProvider(IMemoryCache cache)

{

\_cache = cache;

}

public void Set<T>(string key, T obj)

{

var options = new MemoryCacheEntryOptions();

options.AddExpirationToken(new CancellationChangeToken(\_resetCacheToken.Token));

\_cache.Set(key, obj, options);

}

public T Get<T>(string key)

{

return \_cache.Get<T>(key);

}

public bool TryGetValue<T>(string key, out T obj)

{

if (\_cache.TryGetValue(key, out obj))

{

return true;

}

else

{

return false;

}

}

public void Clean()

{

if (\_resetCacheToken != null && !\_resetCacheToken.IsCancellationRequested && \_resetCacheToken.Token.CanBeCanceled)

{

\_resetCacheToken.Cancel();

\_resetCacheToken.Dispose();

}

\_resetCacheToken = new CancellationTokenSource();

}

}

}

**Листинг файла *Login*.*cshtml*:**

@model VideoRentalWeb.Models.Account.LoginViewModel

<**form** method="post" class="container text-center w-50">

<h2>Login</h2>

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

<p>

<**label** **asp-for**="Login"></**label**><br />

<**input** **asp-for**="Login" class="w-75" />

<p>

<**span** **asp-validation-for**="Login" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" class="w-75" />

<p>

<**span** **asp-validation-for**="Password" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="RememberMe"></**label**>

<**input** **asp-for**="RememberMe" />

</p>

<p>

<input type="submit" value="Login" class="w-75 btn btn-info" />

</p>

</**form**>

**Листинг файла *Logout*.*cshtml*:**

@{

ViewData["Title"] = "Logout";

}

<h2>@ViewData["Title"]</h2>

<**form** **asp-controller**="Account" **asp-action**="Logout" method="post">

@Html.AntiForgeryToken()

<button type="submit">Logout</button>

</**form**>

Листинг файла Program.cs:

@model VideoRentalWeb.Models.Account.RegisterViewModel

@{

ViewData["Title"] = "Register";

}

<**form** method="post" class="container text-center w-50">

<h2>Registration</h2>

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

<p>

<**label** **asp-for**="Login"></**label**><br />

<**input** **asp-for**="Login" class="w-75" />

<p>

<**span** **asp-validation-for**="Login"></**span**>

</p>

<p>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" class="w-75" />

<p>

<**span** **asp-validation-for**="Password"></**span**>

</p>

<p>

<**label** **asp-for**="PasswordConfirm"></**label**><br />

<**input** **asp-for**="PasswordConfirm" class="w-75" />

<p>

<**span** **asp-validation-for**="PasswordConfirm"></**span**>

</p>

<p>

<input type="submit" value="Register" class="w-75 btn btn-info" />

</p>

</**form**>

**Листинг файла *Clientele*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br/>

<**input** **asp-for**="Entity.Description" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Clientele*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Clientele*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br />

<**input** **asp-for**="Entity.Description" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Clientele*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="GenresFilterViewModel.GenreTitle"></**label**>

<input name="DiskName" value="@Model.GenresFilterViewModel.GenreTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Disks*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.DiskViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.CreationYear"></**label**><br/>

<**input** **asp-for**="Entity.CreationYear" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.CreationYear" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Producer"></**label**><br/>

<**input** **asp-for**="Entity.Producer" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.Producer" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.MainActor"></**label**><br/>

<**input** **asp-for**="Entity.MainActor" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.MainActor" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Recording"></**label**><br/>

<**input** **asp-for**="Entity.Recording" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.Recording" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.GenreId"></**label**><br/>

<**input** **asp-for**="Entity.GenreId" required="" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.GenreId" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.DiskType"></**label**><br/>

<**input** **asp-for**="Entity.DiskType" required="" class="w-75" />

<p><**span** **asp-validation-for**="Entity.DiskType" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Disks*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.DiskViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.DiskId" **value**="@Model.Entity.DiskId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Disks*.*Details*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.DiskViewModel

@{

ViewData["Title"] = "Details"; Layout = "\_Layout";

}

<div class="container w-50">

@if (User.Identity.IsAuthenticated)

{

<table class="table table-sm">

<thead class="thead-dark text-center">

<tr>

<th colspan="2">Detailed description of "@Model.Entity.Title"</th>

</tr>

</thead>

<tbody>

<tr>

<td><**label** **asp-for**="Entity.Title"></**label**></td>

<td>@Model.Entity.Title</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.CreationYear"></**label**></td>

<td>@Model.Entity.CreationYear</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.Producer"></**label**></td>

<td>@Model.Entity.Producer</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.MainActor"></**label**></td>

<td>@Model.Entity.MainActor</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.Recording"></**label**></td>

<td>@Model.Entity.Recording</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.GenreId"></**label**></td>

<td>@Model.Entity.GenreId</td>

</tr>

<tr>

<td><**label** **asp-for**="Entity.DiskType"></**label**></td>

<td>@Model.Entity.DiskType</td>

</tr>

</tbody>

</table>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-100">Back</**a**></p>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a></p>

}

</div>

**Листинг файла *Disks*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.DiskViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.DiskId" **value**="@Model.Entity.DiskId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.CreationYear"></**label**><br/>

<**input** **asp-for**="Entity.CreationYear" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.CreationYear" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Producer"></**label**><br/>

<**input** **asp-for**="Entity.Producer" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Producer" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.MainActor"></**label**><br/>

<**input** **asp-for**="Entity.MainActor" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.MainActor" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Recording"></**label**><br/>

<**input** **asp-for**="Entity.Recording" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Recording" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.GenreId"></**label**><br/>

<**input** **asp-for**="Entity.GenreId" required="" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.GenreId" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.DiskType"></**label**><br/>

<**input** **asp-for**="Entity.DiskType" required="" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.DiskType" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Disks*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.DiskViewModel

@{

ViewData["Title"] = "Disks"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="DiskFilterViewModel.DiskTitle"></**label**>

<input name="DiskName" value="@Model.DiskFilterViewModel.DiskTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.DiskTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.DiskId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.DiskId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Genres*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br/>

<**input** **asp-for**="Entity.Description" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Genres*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Genres*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br />

<**input** **asp-for**="Entity.Description" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Genres*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="GenresFilterViewModel.GenreTitle"></**label**>

<input name="DiskName" value="@Model.GenresFilterViewModel.GenreTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Positions*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PositionsViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Positions*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PositionsViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.PositionId" **value**="@Model.Entity.PositionId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Positions*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PositionsViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.PositionId" **value**="@Model.Entity.PositionId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Positions*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="GenresFilterViewModel.GenreTitle"></**label**>

<input name="DiskName" value="@Model.GenresFilterViewModel.GenreTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Pricelist*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PricelistViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.DiskId"></**label**><br/>

<**input** **asp-for**="Entity.DiskId" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.DiskId" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Price"></**label**><br/>

<**input** **asp-for**="Entity.Price" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Price" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Pricelist*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PricelistViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Pricelist*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PricelistViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.PriceId" **value**="@Model.Entity.PriceId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.DiskId"></**label**><br/>

<**input** **asp-for**="Entity.DiskId" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.DiskId" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Price"></**label**><br />

<**input** **asp-for**="Entity.Price" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Price" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Pricelist*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.PricelistViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="PricelistFilterViewModel.Price"></**label**>

<input name="Price" value="@Model.PricelistFilterViewModel.Price"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.DiskTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Price)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Price</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.PriceId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.PriceId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Producers*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.ProducersViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br/>

<**input** **asp-for**="Entity.Description" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Producers*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.ProducersViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.ProduceId" **value**="@Model.Entity.ProduceId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Producers*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.ProducersViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.ProduceId" **value**="@Model.Entity.ProduceId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Manufacturer"></**label**><br/>

<**input** **asp-for**="Entity.Manufacturer" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Manufacturer" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Country"></**label**><br />

<**input** **asp-for**="Entity.Country" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Country" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Producers*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.ProducersViewModel

@{

ViewData["Title"] = "Зкщвгсукы"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<label asp-for="Entities"></label>

</h2>

<div class="container">

<a asp-action="Create" asp-route-sortState="@Model.SortViewModel.CurrentState" asp-route-page="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</a>

<fieldset class="p-2 float-right">

<form method="post" asp-route-page="@Model.PageViewModel.CurrentPage">

<label asp-for="ProducersFilterViewModel.Manufacturer"></label>

<input name="Manufacturer" value="@Model.ProducersFilterViewModel.Manufacturer"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</form>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.DiskProducerSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Manufacturer)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Manufacturer</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.ProduceId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.ProduceId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Roles*.*Create*.*cshtml*:**

@model RoleViewModel

<**form** method="post" class="container text-center w-50">

<h2>Create</h2>

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

<p>

<**label** **asp-for**="RoleName"></**label**><br />

<**input** **asp-for**="RoleName" class="w-75" />

<p>

<**span** **asp-validation-for**="RoleName"></**span**>

</p>

<p>

<input type="submit" value="Create" class="w-75 btn btn-success" />

</p>

<p>

<**a** **asp-action**="Index" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

**Листинг файла *Roles*.*Edit*.*cshtml*:**

@using Microsoft.AspNetCore.Identity

@model RoleViewModel

<**form** **asp-action**="Edit" method="post" class="container text-center w-25">

<input type="hidden" name="userId" value="@Model.User.Id" />

<div class="form-group">

@foreach (IdentityRole role in Model.Roles)

{

<input type="checkbox" name="roles" value="@role.Name" class="float-left"

@(Model.UserRoles.Contains(role.Name) ? "checked=\"checked\"" : "") />@role.Name <br />

}

</div>

<button type="submit" class="btn btn-primary w-100">Сохранить</button>

</**form**>

**Листинг файла *Roles*.*Index*.*cshtml*:**

@model RoleViewModel

<div class="container text-center w-50">

<h2>

<**label** **asp-for**="Roles"></**label**>

</h2>

<p>

<div class="container text-left">

<**a** **asp-action**="Create" class="btn btn-primary">Add a new role</**a**>

<**a** **asp-action**="Users" class="btn btn-primary">User list</**a**>

</div>

</p>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center">

<tr>

<th class="w-50"><**label** **asp-for**="RoleName"></**label**></th>

<th class="w-25"><**label** **asp-for**="Modification"></**label**></th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Roles)

{

<tr>

<td>@entity.Name</td>

<td class="text-center">

<**form** **asp-action**="Delete" **asp-route-name**="@entity.Name" method="post">

<button type="submit" class="btn btn-danger">Delete</button>

</**form**>

</td>

</tr>

}

</tbody>

</table>

</div>

**Листинг файла *Roles*.*Users*.*cshtml*:**

@model IEnumerable<VideoRentalWeb.DataModels.User>

<div class="container text-center w-50">

<h2>User list</h2>

<table class="table">

@foreach (var user in Model)

{

<tr>

<td>@user.Email</td>

<td>

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-userid**="@user.Id">Manage acces</**a**>

</td>

</tr>

}

</table>

</div>

**Листинг файла *Staff*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.StaffViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br/>

<**input** **asp-for**="Entity.Description" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Staff*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Staff*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.StaffViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br />

<**input** **asp-for**="Entity.Description" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Staff*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.StaffViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="GenresFilterViewModel.GenreTitle"></**label**>

<input name="DiskName" value="@Model.GenresFilterViewModel.GenreTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Takings*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Create"; Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br/>

<**input** **asp-for**="Entity.Description" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Takings*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**> }

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Takings*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.GenreId" **value**="@Model.Entity.GenreId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p>

<**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**>

</p>

<p>

<**label** **asp-for**="Entity.Description"></**label**><br />

<**input** **asp-for**="Entity.Description" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Description" class="text-danger"></**span**>

</p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p>

<**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

}

else

{

<p>You are not logged in. Please

<a href="/Account/Login">login</a>.

</p>

}</div>

**Листинг файла *Takings*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.GenreViewModel

@{

ViewData["Title"] = "Genres"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="GenresFilterViewModel.GenreTitle"></**label**>

<input name="DiskName" value="@Model.GenresFilterViewModel.GenreTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.GenreId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Types*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.TypeViewModel;

@{

ViewData["Title"] = "Create";

Layout = "\_Layout";

}

<div class="container text-center w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<h2>Create a new entity</h2>

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</div>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.TypeId"></**label**><br/>

<**input** **asp-for**="Entity.TypeId" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Add" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

} else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Types*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.TypeViewModel

@{

ViewData["Title"] = "Delete"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-75">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<p>

<**input** hidden **asp-for**="Entity.TypeId" **value**="@Model.Entity.TypeId"/>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

</p>

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p>

<h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**>

} else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Types*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.TypeViewModel

@{

ViewData["Title"] = "Edit"; Layout = "\_Layout";

}

<div class="container text-center align-middle w-50">

@if (User.Identity.IsAuthenticated)

{

<**form** method="post">

<div>

<**input** hidden **asp-for**="PageViewModel.CurrentPage" **value**="@Model.PageViewModel.CurrentPage"/>

<**input** hidden **asp-for**="Entity.TypeId" **value**="@Model.Entity.TypeId"/>

</div>

<h2>Edit entity</h2>

<**div** **asp-validation-summary**="ModelOnly" class="text-danger text-justify w-100 m-lg-0"></**div**>

<p>

<**label** **asp-for**="Entity.Title"></**label**><br/>

<**input** **asp-for**="Entity.Title" class="w-75"/>

<p><**span** **asp-validation-for**="Entity.Title" class="text-danger"></**span**></p>

<p>

<input type="submit" value="Save" class="btn btn-success w-75"/>

<p><**a** **asp-action**="Index" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger w-75">Back</**a**></p>

</**form**>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Types*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.TypeViewModel

@{

ViewData["Title"] = "Types"; Layout = "\_Layout";

}

<div class="container w-100">

@if (User.Identity.IsAuthenticated)

{

<h2 class="text-center">

<**label** **asp-for**="Entities"></**label**>

</h2>

<div class="container">

<**a** **asp-action**="Create" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-primary p-2">Add new entity</**a**>

<fieldset class="p-2 float-right">

<**form** method="post" **asp-route-page**="@Model.PageViewModel.CurrentPage">

<**label** **asp-for**="TypeFilterViewModel.TypeTitle"></**label**>

<input name="DiskName" value="@Model.TypeFilterViewModel.TypeTitle"/>

<input type="submit" value="Filter" class="btn btn-primary"/>

</**form**>

</fieldset>

</div>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center text-white">

<tr>

<th class="w-25">

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.GenreTitleSort" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="nav-link text-white">

@Html.DisplayNameFor(e => e.Entity.Title)

</**a**>

</th>

<th class="w-25 align-middle">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.Title</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.TypeId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.TypeId" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@Model.PageViewModel.CurrentPage" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

<div class="container">

<div class="float-left w-50">

<**form** method="get">

<input type="number" name="page" min="1" max="@Model.PageViewModel.TotalPages" value="@Model.PageViewModel.CurrentPage" class="w-25"/>

<input type="submit" value="Move" class="btn btn-primary"/>

</**form**>

</div>

<div class="text-right float-right">

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage - 1)" class="btn btn-primary">Back</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index" **asp-route-sortState**="@Model.SortViewModel.CurrentState" **asp-route-page**="@(Model.PageViewModel.CurrentPage + 1)" class="btn btn-primary">Next</**a**>

}

</div>

</div>

}

else

{

<p>You are not logged in. Please <a href="/Account/Login">login</a>.</p>

}

</div>

**Листинг файла *Users*.*ChangePassword*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.UsersViewModel

<**form** method="post" class="container text-center w-50">

<h2>Change password</h2>

<p>

<**input** **asp-for**="Entity.Id" **value**="@Model.Entity.Id" hidden />

</p>

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

<p>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" class="w-75" />

<p>

<**span** **asp-validation-for**="Password"></**span**>

</p>

<p>

<**label** **asp-for**="NewPassword"></**label**><br />

<**input** **asp-for**="NewPassword" class="w-75" />

<p>

<**span** **asp-validation-for**="NewPassword"></**span**>

</p>

<p>

<input type="submit" value="Change" class="w-75 btn btn-success" />

</p>

<p>

<**a** **asp-action**="Index" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

**Листинг файла *Users*.*Create*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.UsersViewModel

<**form** method="post" class="container text-center w-50">

<h2>Create</h2>

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

<p>

<**label** **asp-for**="Entity.Email"></**label**><br />

<**input** **asp-for**="Entity.Email" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Email"></**span**>

</p>

<p>

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" class="w-75" />

<p>

<**span** **asp-validation-for**="Password"></**span**>

</p>

<p>

<input type="submit" value="Create" class="w-75 btn btn-success" />

</p>

<p>

<**a** **asp-action**="Index" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

**Листинг файла *Users*.*Delete*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.UsersViewModel

<div class="container text-center align-middle w-75">

<**form** method="post">

<p>

<**input** **asp-for**="Entity.Id" **value**="@Model.Entity.Id" hidden />

<**input** **asp-for**="DeleteViewModel.IsDeleted" **value**="@Model.DeleteViewModel.IsDeleted" hidden />

</p>

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

@if (!Model.DeleteViewModel.IsDeleted)

{

<h2>Deletion process</h2>

<p><h3>@Model.DeleteViewModel.Message</h3></p>

<p><input type="submit" class="btn btn-danger w-50" value="Delete"></p>

<p><**a** **asp-action**="Index" class="btn btn-primary w-50">Back</**a**></p>

}

else

{

<h2>Deletion result</h2>

<p><h3>@Model.DeleteViewModel.Message</h3></p>

<p><**a** **asp-action**="Index" class="btn btn-primary w-50">Back</**a**></p>

}

</**form**>

</div>

**Листинг файла *Users*.*Edit*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.UsersViewModel

<**form** method="post" class="container text-center w-50">

<h2>Edit</h2>

<p>

<**input** **asp-for**="Entity.Id" **value**="@Model.Entity.Id" hidden />

</p>

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

<p>

<**label** **asp-for**="Entity.Email"></**label**><br />

<**input** **asp-for**="Entity.Email" **value**="@Model.Entity.Email" class="w-75" />

<p>

<**span** **asp-validation-for**="Entity.Email"></**span**>

</p>

<p>

<input type="submit" value="Edit" class="w-75 btn btn-success" />

</p>

<p>

<**a** **asp-action**="Index" class="btn btn-danger w-75">Back</**a**>

</p>

</**form**>

**Листинг файла *Users*.*Index*.*cshtml*:**

@model VideoRentalWeb.Models.Entities.UsersViewModel

<div class="container text-center">

<h2>

<**label** **asp-for**="Entities"></**label**>

</h2>

<p>

<div class="container text-left">

<**a** **asp-action**="Create" class="btn btn-primary">Add a new user</**a**>

</div>

</p>

<table class="table table-hover table-sm">

<thead class="thead-dark text-center">

<tr>

<th class="w-50"><**label** **asp-for**="Entity.Email"></**label**></th>

<th class="w-25">Modification</th>

</tr>

</thead>

<tbody>

@foreach (var entity in Model.Entities)

{

<tr>

<td>@entity.UserName</td>

<td class="text-center">

<**a** **asp-action**="Edit" **asp-route-id**="@entity.Id" class="btn btn-warning">Edit</**a**>

<**a** **asp-action**="ChangePassword" **asp-route-id**="@entity.Id" class="btn btn-warning">Change password</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@entity.Id" class="btn btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

</div>

**Листинг файла *\_Layout*.*cshtml*:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Video rental</title>

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<**a** class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Video rental</**a**>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Home</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Disks" **asp-action**="Index">Disks</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Genres" **asp-action**="Index">Genres</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Types" **asp-action**="Index">Types</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="Login">Login</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="Register">Register</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="Logout">Logout</**a**>

</li>

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2023 - Video rental - <**a** **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</**a**>

</div>

</footer>

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** **src**="~/js/site.js" **asp-append-version**="true"></**script**>

@RenderSection("Scripts", required: false)

</body>

</html>

# ПРИЛОЖЕНИЕ Б

(обязательное)

**Чертёж структуры *web*-приложения**