СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная курсовой проект посвящён разработке *web*-приложения баз данных выпуска и реализации сходных видов продукции, создание интерфейса в виде набора *web*-страниц, обеспечивающих отображение и редактирование информации из базы данных, для автоматизации работы со структурированной информацией.

Наиболее используемым типом информационной системы является клиент-серверная система. Данный тип системы представляет собой взаимодействие структурных компонентов, где структурными компонентами являются сервер и узлы-поставщики определённых сервисов, а также клиенты, которые пользуются данным сервисом. Данный тип системы наиболее часто используется в создании корпоративных баз данных, в которой 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 данной задачи определены следующие исходные данные:

– предприятия (наименование, ФИО руководителя, вид деятельности, форма собственности);

– виды продукции;

– продукция (наименование, характеристики продукции, единица измерения вида продукции, фото);

– план выпуска и фактический выпуск продукции (предприятие, наименование вида продукции, планируемый объем выпуска, фактический объем выпуска, квартал, год);

– план реализации и фактическая реализация продукции (предприятие, наименование вида продукции, планируемый объем реализации, фактический объем реализации, квартал, год);

Также прилагаются следующие дополнительные требования к отображению данных:

– сравнение одинаковых видов продукции разных предприятий по характеристикам;

– анализ выполнения плана выпуска продукции (вывод отношения «факт / план» в процентном выражении для всех видов продукции всех предприятий за заданный квартал текущего года; отображение списка предприятий, не выполнивших план более чем по двум видам продукции за определенный квартал заданного года);

– Анализ выполнения плана реализации продукции (вывод отношения «(факт – план)/план» в процентном выражении для всех видов продукции определенного предприятия за все кварталы заданного года; отображение списка предприятий и их руководителей, выполнивших план реализации по всем видам продукции за последние два года; определение предприятия с максимальным средним процентом выполнения плана реализации по всем видам продукции за текущий квартал).

Для решения поставленной задачи в качестве СУБД используется *MS SQL Server*. Данная 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*. Данная 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:

– «Предприятия»;
– «Виды продукции»;
– «Продукция»;
– «Реализация»;
– «Выпуск».

Для сущности «Предприятия» было создано отношение (таблица) с атрибутами: «Код предприятия», «Наименование», «ФИО руководителя», «Вид деятельности», «Форма собственности». Подробное описание отношения и атрибутов приведено в таблице 1.1. Данное отношение находится в первой нормальной форме.

Таблица 1.1 – **Отношение описывающие сущность «Предприятие»**

Атрибуты	**Описание домена**	**Тип данных**
Код предприятия	Уникальный инкрементируемый идентификатор для каждого предприятия. Является первичным ключом.	Целое число
Наименования	Содержит наименования предприятий.	Строка
ФИО руководителя	Содержит ФИО руководителей предприятий.	Строка
Вид деятельности	Содержит описание видов деятельности предприятий.	Строка
Форма собственности	Содержит описание форм собственности предприятий.	Строка

Отношение для сущности «Реализация», описано в таблице 1.2. Отношение по условию задачи должно содержать атрибуты: «Код реализации», «Предприятие», «Наименование вида продукции», «Планируемый объем реализации», «Фактический объем реализации», «Квартал», «Год». Данное отношение следует привести к третей нормальной форме, заменив атрибут «Наименование вида продукции» на атрибут «Код вида продукции» связав отношение «Реализация» с отношением «Вид продукции», а также заменив атрибут «Предприятие» на атрибут «Код предприятия», связав отношение «Реализация» с отношение «Предприятия».

Таблица 1.2 – **Отношение описывающие сущность «Реализация»**

Атрибуты	**Описание домена**	**Тип данных**
Код реализации	Уникальный инкрементируемый идентификатор для каждой реализации. Является первичным ключом.	Целое число
Планируемый объем реализации	Содержит числовое значение планируемого объема реализации.	Целое число
Фактический объем реализации	Содержит числовое значение фактического объема реализации.	Целое число
Квартал	Содержит числовое значение квартала.	Целое число
Год	Содержит числовое значение года.	Целое число
Код предприятия	Содержит ссылки на код предприятий. Является внешним ключом для связи с отношением «Предприятия».	Целое число
Код вида продукции	Содержит ссылки на код вида продукции. Является внешним ключом для связи с отношением «Виды продукции».	Целое число

Отношение для сущности «Выпуск», описано в таблице 1.3. Отношение по условию задачи должно содержать атрибуты: «Код выпуска», «Предприятие», «Наименование вида продукции», «Планируемый объем выпуска», «Фактический объем выпуска», «Квартал», «Год». Данное отношение следует привести к третей нормальной форме, заменив атрибут «Наименование вида продукции» на атрибут «Код вида продукции» связав отношение «Выпусе» с отношением «Вид продукции», а также заменив атрибут «Предприятие» на атрибут «Код предприятия», связав отношение «Выпуск» с отношение «Предприятия».

Таблица 1.3 – **Отношение описывающие сущность «Выпуск»**

Атрибуты	**Описание домена**	**Тип данных**
Код выпуска	Уникальный инкрементируемый идентификатор для каждого выпуска продукции. Является первичным ключом.	Целое число

Продолжение таблицы 1.3

Планируемый объем выпуска	Содержит числовое значение планируемого объема выпуска.	Целое число
Фактический объем выпуска	Содержит числовое значение фактического объема выпуска.	Целое число
Квартал	Содержит числовое значение квартала.	Целое число
Год	Содержит числовое значение года.	Целое число
Код предприятия	Содержит ссылки на код предприятий. Является внешним ключом для связи с отношением «Предприятия».	Целое число
Код вида продукции	Содержит ссылки на код вида продукции. Является внешним ключом для связи с отношением «Виды продукции».	Целое число

Отношение для «Видов продукции», таблица 1.4, состоит из атрибутов: «Код вида продукции», «Наименование», «Код продукции».

Таблица 1.4 – **Отношение описывающие сущность «Вид продукции»**

Атрибуты	**Описание домена**	**Тип данных**
Код вида неисправности	Уникальный инкрементируемый идентификатор для каждой неисправности. Является первичным ключом.	Целое число
Код модели оборудования	Содержит код ремонтируемой модели. Является внешним ключом для связи с отношением «Ремонтируемые модели».	Целое число
Наименование	Содержит наименование ремонтируемых моделей.	Строка
Методы ремонта	Содержит описывание методов ремонта неисправностей.	Строка
Цена работы	Содержит стоимость работы для устранения неисправности. Вычисляется относительно стоимости запчастей.	Число с плавающей точкой
Код вида неисправности	Содержит код вида неисправности. Является внешним ключом для связи с отношением «Виды неисправностей».	Целое число

Сущность «Продукция», таблица 1.5, состоит из атрибутов: «Код продукции», «Наименование», «Единица измерения вида продукции», «Характеристики продукции», «Фото», «Код вида продукции».

Таблица 1.5 – **Отношение описывающие сущность «Продукция»**

Атрибуты	**Описание домена**	**Тип данных**
Код продукции	Уникальный инкрементируемый идентификатор для каждой продукции. Является первичным ключом.	Целое число
Наименование	Содержит наименование продукции.	Строка
Единица измерения вида продукции	Содержит единицу измерения вида продукции	Строка
Характеристики продукции	Содержит характеристики продукции	Строка
Фото	Содержит фото продукции	Изображение
Код вида продукции	Уникальный инкрементируемый идентификатор для вида продукции. Является внешним ключом для связи с отношением «Виды продукции».	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 соответствующие 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*.

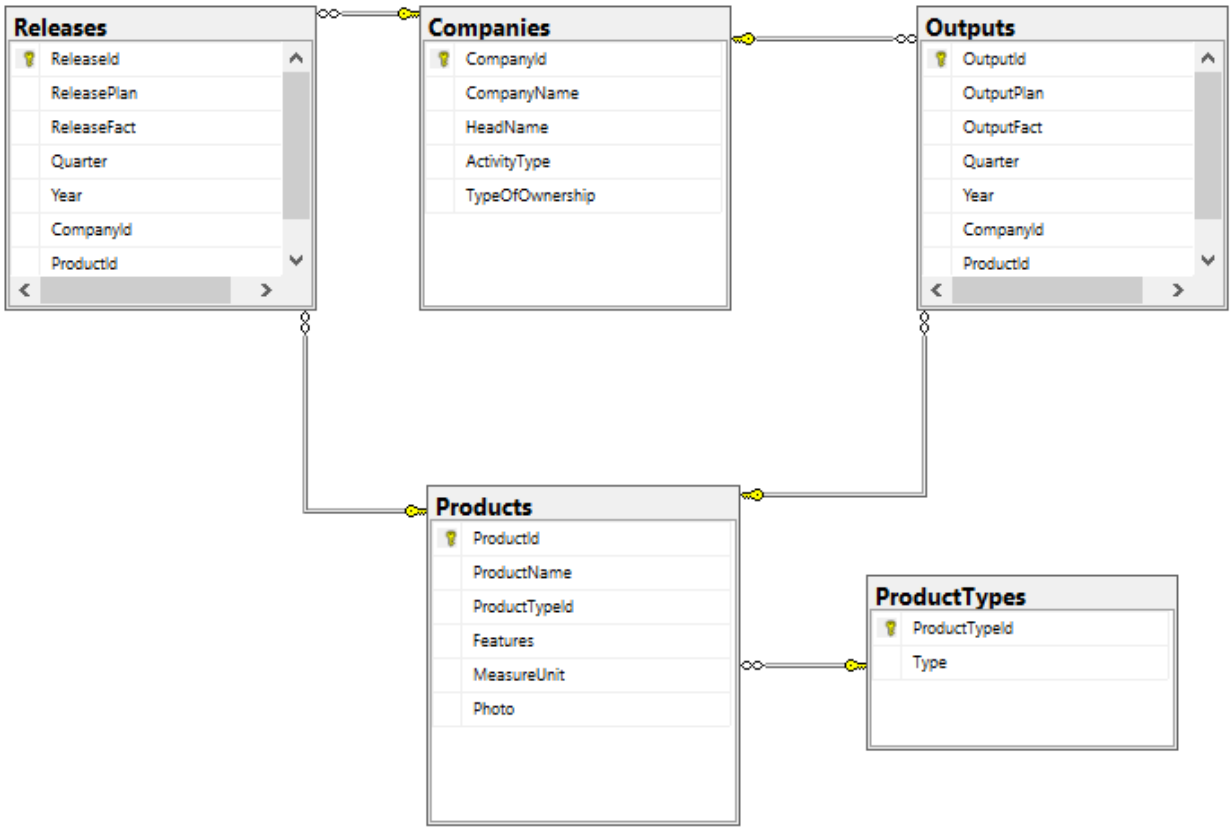

Рисунок 1.1 – Диаграмма базы данных

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: *Companies, Products, ProductTypes, Outputs, Releases.*

Таблицы *Companies, Products, ProductTypes* находятся в отношении «один» и описывают сущности «Предприятия», «Продукция» и «Виды продукции» и подобраны физические тип данных для соответствующих столбцов, установлены первичные ключи.

Таблицы *Outputs, Releases* находятся в отношении «многие», описывают сущности «Выпуск», «Реализация». 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 ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Требования к системному и прикладному программному обеспечению на стороне сервера хранилища данных

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.

2.2 Требования к системному и прикладному программному обеспечению на стороне *web*-сервера

Минимальные требования к аппаратному и программному обеспечению и корректной работы на нём, необходимо соблюдение следующих условий:

– 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 приложение разработана на платформе *.NET Core*, 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.

2.3 Требования к системному и прикладному программному обеспечению на стороне клиента

Чтобы приложение корректно работало на стороне клиента требуется браузера с поддержкой «*Bootstrap*» и наличие клиента и *web*-сервера в одной сети (локальной, глобальной).

2.4 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/netcoreapp2.1/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.

Чтобы подключиться 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 данного 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. Данный 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 *Companies, Products, ProductTypes, Outputs, Releases* описывают таблицы *Companies, Products, ProductTypes, Outputs, Releases* соответственно. Код данных классов представлен в приложении Б.

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*. Данный 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 патерна *{controller=[ControllerName]}/{action=[MethodName]}*, где [*ControllerName*] – название контроллера, [*MethodName*] – название метода контроллера.

Для работы с созданными моделями разработаны следующие контроллеры:

– *HomeController* – отвечает за вывод начальной страницы;

– *CompaniesController* – отвечает за работу с таблицей *Companies;*

– *ProductsControllers* – отвечает за работу с таблицей *Products;*

– *ProductTypesController* – отвечает за работу с таблицей *ProductTypes;*

– *OutputsController* – отвечает за работу с таблицей *Outputs;*

– *ReleasesController* – отвечает за работу с таблицей *Releases.*

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:

– *Companies* – содержит представления для работы с данными о предприятиях, для данного представления был создан дополнительный класс «модель-представление», с помощью которого можно передавать несколько объектов представлению;

– *Products* – содержит представления для работы с данными о продукции сотрудников;

– *ProductTypes* – содержит представления для работы с данными о видах продукции;

– *Outputs* – содержит представления для работы с таблицей «Выпуск»;

– *Releases* – содержит представления для работы с таблицей «Реализация».

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). Так же некоторые данные выборки, которые редко редактируются и добавляются, 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].

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Введение

Данное *web*-приложение производит автоматизацию процесса выпуска и реализации сходных видов продукции.

4.2 Назначение, условие применения и функционал

Web-приложение предназначено для управления и учёта данных о выпуске и реализации сходных видов продукции предприятиями.

Основные функции приложения:
– добавление, просмотр и редактирования предприятий;
– выборка, сортировка и фильтрация предприятий по заданным критериям;
– добавление, просмотр и редактирования данных о видах продукции;
– добавление, просмотр и редактирования данных о продукции;
– фильтрация, сортировка и выборка продукции по заданным критериям;
– добавление, просмотр и редактирования данных о выпуске сходных видов продукции;
– добавление, просмотр и редактирования данных о реализации сходных видов продукции;
– автоматизация рабочего процесса.

4.3 Подготовка к работе

Для использования приложение требуется веб-браузер (*Mozilla Firefox*, *Chrome*, *Opera*, *Microsoft Edge* и пр.) в адресной строке веб-браузера ввести *URL*-адрес выданный системным-администратором и нахождение устройства в той же локальной сети, где находится *web*-сервер (если сервер находится в глобальной сети, то подключение к интернету).

4.4 Описание операции по обработки данных

Для операции просмотра данных о предприятиях, требуется выбрать вкладку «Предприятия» в вверху окна браузера, рисунок 4.1.

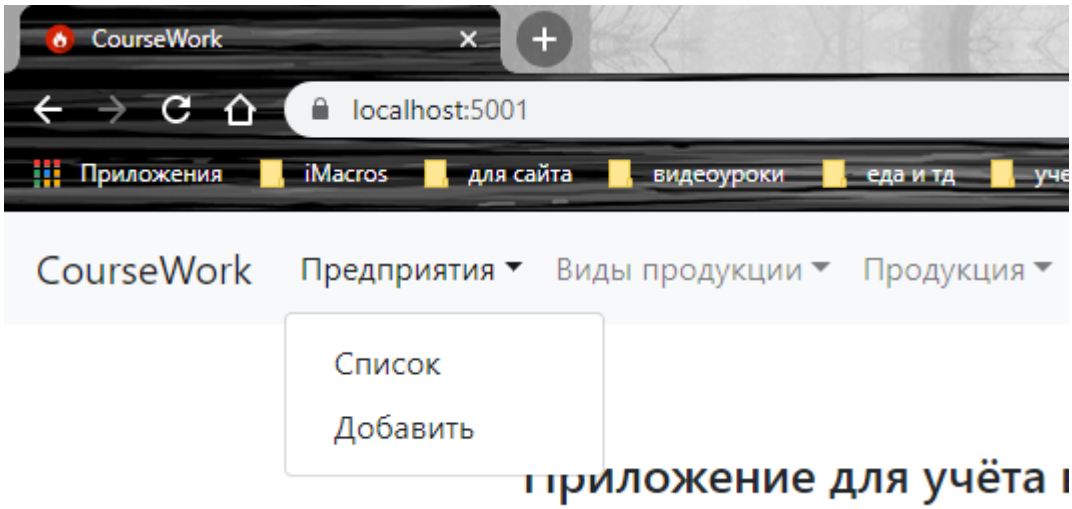

Рисунок 4.1 – Выбор предприятий

Затем выбрать вкладку «Список» и загрузится новое окно со списком предприятий, рисунок 4.2.

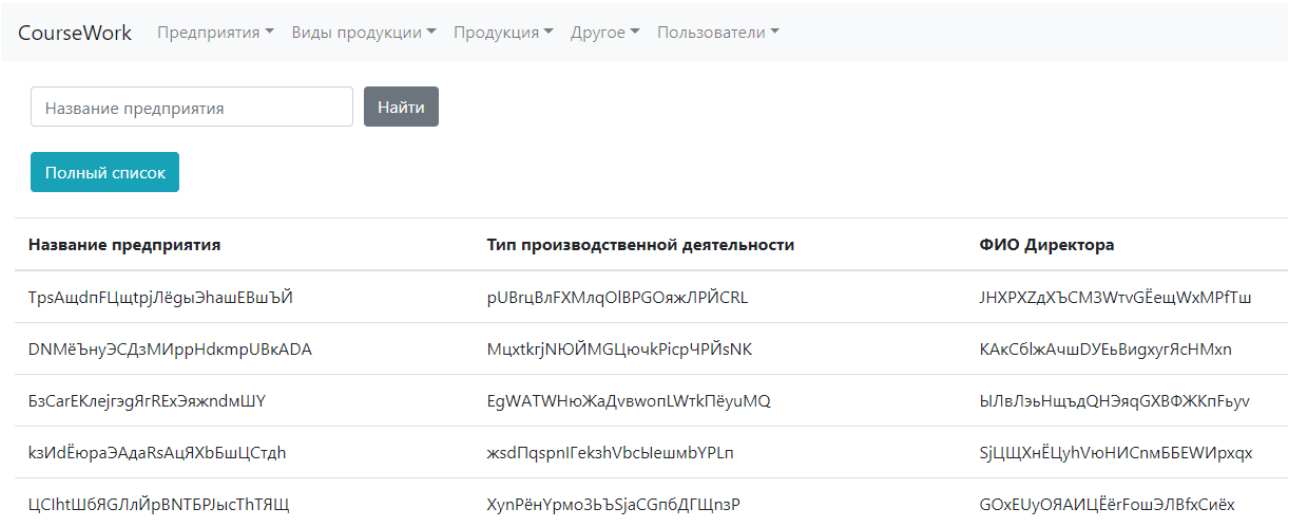

Рисунок 4.2 – Список предприятий по выпуску и реализации продукции

Так же можно осуществлять поиск по названию предприятия.

Чтобы добавить новое предприятие, требуется выбрать вкладку «Предприятия» в вверху окна браузера, затем выбрать вкладку «Добавить» и загрузится новое окно с формами для добавления предприятия (все поля добавить предприятие, нужно нажать на кнопку «Добавить».

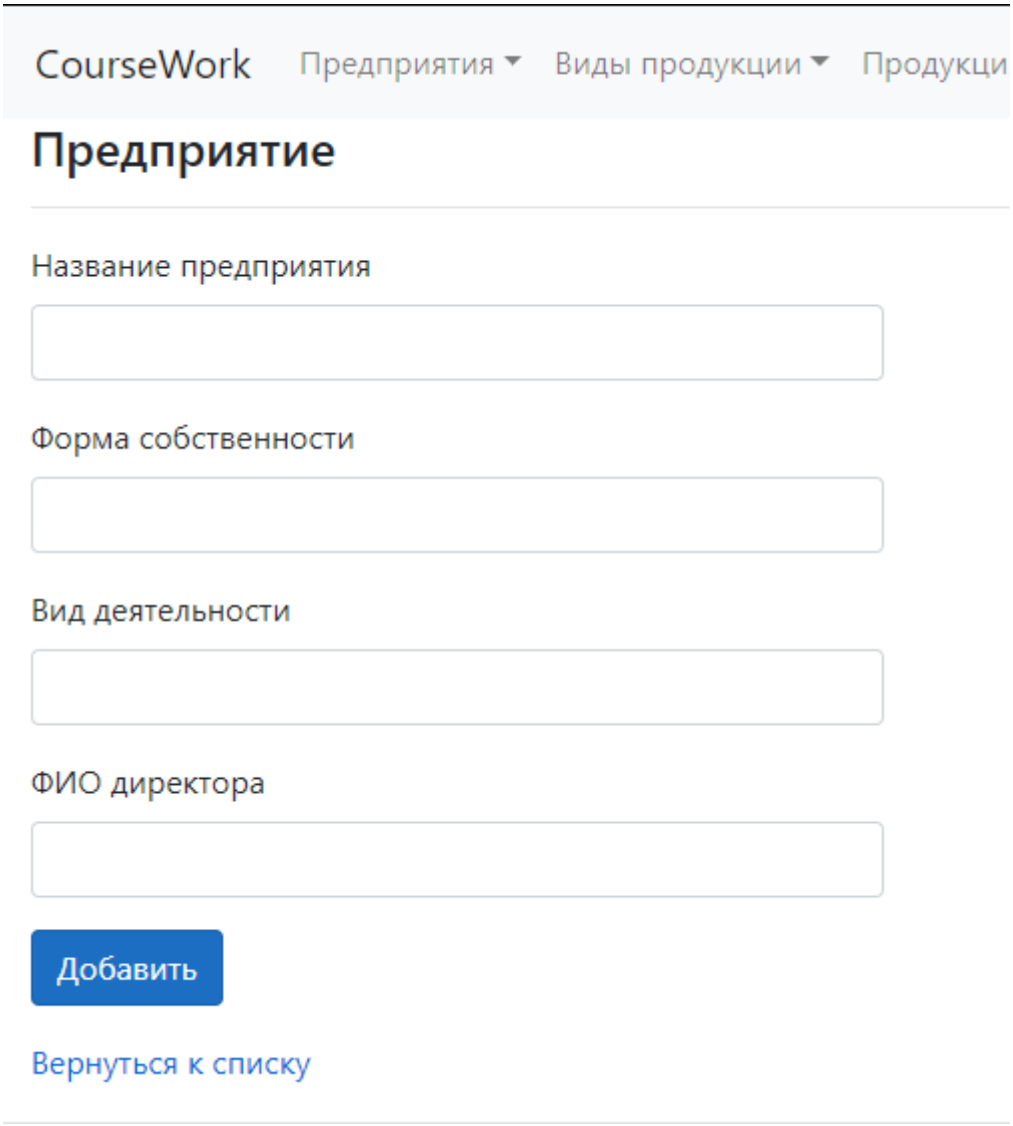

Рисунок 4.3 – Форма для добавления нового предприятия

После добавления предприятия пользователь будет перенаправлен на страницу с выборкой.

По той же аналогии, описанной выше, можно производить аналогичные манипуляции с данными других сервисов.

5 РУКОВОДСТВО ПРОГРАММИСТА

5.1 Назначения и условия применения программы

Приложение предназначена для предоставления информации из базы данных предприятиям по ремонту оборудования, чтобы автоматизировать учёт заказов по их ремонту 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 2019*.

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 выпуска и реализации сходных видов продукции. 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 является вполне достаточным для выполнения основных задач, и структура спроектирована таким образом, что его дальнейшее расширение не приведёт ни к каким трудностям: изменению структуры или переписыванию логики. Все вышеперечисленные преимущества, поможет мелким сервисам по ремонту оборудования автоматизироваться свой производственный 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 Core* 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. https://docs.microsoft.com/ru-ru/ASPNET/core/host-and-deploy/?view=aspnetcore-2.1

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. 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.

5. ASP.NET Core. Dependency Injection, Электрон. данные. – Режим доступа: https://metanit.com/sharp/aspnet5/6.1.php. Дата доступа: 13.12.2019.

6. 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

Appsettings.json:
```
{
  "ConnectionStrings": {
    "SQLServerConnection":
"Server=GLOBEELE\\GLOBEELE;Database=RP1;Trusted_Connection=True;MultipleActiveResultSets=Tru
e",
    "UsersSQLServerConnection":
"Server=GLOBEELE\\GLOBEELE;Database=users;Trusted_Connection=True;MultipleActiveResultSets=t
rue"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

AccountController.cs:
```
using System.Threading.Tasks;
using CourseWorkDb.Models.Authentication;
using CourseWorkDb.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace CourseWorkDb.Controllers
{
    [AllowAnonymous]
    public class AccountController : Controller
    {
        private readonly SignInManager<User> _signInManager;

        public AccountController(SignInManager<User> signInManager)
        {
            _signInManager = signInManager;
        }

        [HttpGet]
        public IActionResult Login(string returnUrl = null)
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return View(new LoginViewModel { ReturnUrl = returnUrl });
            }
        }

        [HttpGet]
        public IActionResult AccessDenied(string returnUrl = null)
```

```
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return View(new LoginViewModel { ReturnUrl = returnUrl });
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (ModelState.IsValid)
            {
                Microsoft.AspNetCore.Identity.SignInResult result = await _signInMan-
ager.PasswordSignInAsync(model.Email, model.Password, false, false);
                if (result.Succeeded)
                {
                    if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.Re-
turnUrl))
                    {
                        return Redirect(model.ReturnUrl);
                    }
                    else
                    {
                        return RedirectToAction("Index", "Home");
                    }
                }
                else
                {
                    ModelState.AddModelError("", "Неправильный логин или пароль");
                }
            }
            return View(model);
        }

        [Authorize]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> LogOff()
        {
            await _signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");
        }
    }
                                            }
```

CompaniesController.cs:

```
namespace CourseWorkDb.Controllers
{
    [Authorize]
    public class CompaniesController : Controller
    {
        private const int PageSize = 10;
        private const string SessionKey1 = "viol_company_filter";
        private const string SessionKey2 = "viol_year_filter";
        private const string SessionKey3 = "cmpr_company_filter";
        private const string SessionKey4 = "cmpr_year_filter";

        private readonly RationingDbContext _context;
```

```
        public CompaniesController(RationingDbContext context)
        {
            _context = context;
        }

        // GET: Companies
        public async Task<IActionResult> Index(string searchByName, int page = 1)
        {
            ViewData["CurrentSearch"] = searchByName;

            IQueryable<Company> source = _context.Companies;

            if (!string.IsNullOrEmpty(searchByName))
            {
                source = source.Where(item => item.CompanyName.Contains(searchByName));
            }

            List<Company> list = await source.Skip((page - 1) *
PageSize).Take(PageSize).ToListAsync();

            IndexViewModel<Company> viewModel = new IndexViewModel<Company>
            {
                PageViewModel = new PageViewModel(await source.CountAsync(), page,
PageSize),
                Items = list
            };
            return View(viewModel);
        }

        // GET: Companies/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var company = await _context.Companies
                .FirstOrDefaultAsync(m => m.CompanyId == id);
            if (company == null)
            {
                return NotFound();
            }

            return View(company);
        }

        // GET: Companies/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Companies/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,CompanyName,FormOfOwnership,Head-
Name,ActivityType")] Company company)
        {
            if (ModelState.IsValid)
            {
                _context.Add(company);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
```

```
            }
            return View(company);
        }

        // POST: Companies/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,CompanyName,FormOfOwner-
ship,HeadName,ActivityType")] Company company)
        {
            if (id != company.CompanyId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(company);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!CompanyExists(company.CompanyId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(company);
        }

        // POST: Companies/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var company = await _context.Companies.FindAsync(id);
            _context.Companies.Remove(company);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool CompanyExists(int id)
        {
            return _context.Companies.Any(e => e.CompanyId == id);
        }

        // GET: Companies/Violators
        public async Task<IActionResult> Violators(int? company, int? year, int page = 1)
        {
            if (company == null)
            {
                company = HttpContext.Session.GetInt32(SessionKey1);
            }

            if (year == null)
            {
```

```
                year = HttpContext.Session.GetInt32(SessionKey2);
            }

            var list = _context.Outputs.Include(item => item.Company).Include(item =>
item.Product)
                .Join(_context.Releases, a => new { a.Year, a.Quarter, a.CompanyId,
a.ProductId }, b => new { b.Year, b.Quarter, b.CompanyId, b.ProductId },
                (a, b) => new
                {
                    a.CompanyId,
                    a.Company.CompanyName,
                    a.Year,
                    a.Quarter,
                    a.Product.ProductName,
                    a.Product.MeasureUnit,
                    a.Product.ProductType,
                    Excess = (b.ReleaseFact / b.ReleasePlan) - a.OutputFact
                }).Where(item => item.Excess > 0.0f);

            if (company != null)
            {
                HttpContext.Session.SetInt32(SessionKey1, company.Value);
                if (company != 0)
                {
                    list = list.Where(p => p.CompanyId == company);
                }
            }

            if (year != null)
            {
                HttpContext.Session.SetInt32(SessionKey2, year.Value);
                if (year != 0)
                {
                    list = list.Where(p => p.Year == year);
                }
            }

            int count = list.Count();

            List<ComparisonViewModel> itemList = await list.Skip((page - 1) *
PageSize).Take(PageSize)
                .Select(item => new ComparisonViewModel
                {
                    CompanyId = item.CompanyId,
                    CompanyName = item.CompanyName,
                    ProductName = item.ProductName,
                    MeasureUnit = item.MeasureUnit,
                    Year = item.Year,
                    Quarter = item.Quarter,
                    Excess = item.Excess
                }).ToListAsync();

            IndexViewModel<ComparisonViewModel> viewModel = new IndexViewModel<Compari-
sonViewModel>
            {
                PageViewModel = new PageViewModel(count, page, PageSize),
                FilterViewModel = new FilterViewModel(_context.Companies.ToList(), company,
                    _context.Outputs.GroupBy(item => item.Year).Select(item => new Filter-
ViewModel.Year
                    {
                        Name = item.Key.ToString(),
                        Number = item.Key
                    }).OrderBy(item => item.Number).ToList(), year),
                Items = itemList
```

```
            };
            return View(viewModel);
        }

        // GET: Companies/Comparison
        public async Task<IActionResult> Comparison(int? company, int? year, int page = 1)
        {
            if (company == null)
            {
                company = HttpContext.Session.GetInt32(SessionKey3);
            }

            if (year == null)
            {
                year = HttpContext.Session.GetInt32(SessionKey4);
            }

            var list = _context.Releases.Include(item => item.Company).Include(item =>
item.Product)
                .Join(_context.Outputs, a => new { a.Year, a.Quarter, a.CompanyId,
a.ProductId }, b => new { b.Year, b.Quarter, b.CompanyId, b.ProductId },
                (a, b) => new
                {
                    a.CompanyId,
                    a.Company.CompanyName,
                    a.Product.ProductName,
                    a.Product.ProductType,
                    a.Product.MeasureUnit,
                    a.Year,
                    a.Quarter,
                    a.ReleasePlan,
                    a.ReleaseFact,
                    OutputDifference = b.OutputFact / b.OutputPlan
                });

            if (company != null)
            {
                HttpContext.Session.SetInt32(SessionKey3, company.Value);
                if (company != 0)
                {
                    list = list.Where(p => p.CompanyId == company);
                }
            }

            if (year != null)
            {
                HttpContext.Session.SetInt32(SessionKey4, year.Value);
                if (year != 0)
                {
                    list = list.Where(p => p.Year == year);
                }
            }

            int count = list.Count();

            List<ComparisonViewModel> itemList = await list.Skip((page - 1) *
PageSize).Take(PageSize)
                .Select(item => new ComparisonViewModel
                {
                    CompanyId = item.CompanyId,
                    CompanyName = item.CompanyName,
                    MeasureUnit = item.MeasureUnit,
                    ProductName = item.ProductName,
                    Year = item.Year,
```

```
                    Quarter = item.Quarter,
                    ReleaseFact = item.ReleaseFact,
                    ReleasePlan = item.ReleasePlan
                }).ToListAsync();

            IndexViewModel<ComparisonViewModel> viewModel = new IndexViewModel<Compari-
sonViewModel>
            {
                PageViewModel = new PageViewModel(count, page, PageSize),
                FilterViewModel = new FilterViewModel(_context.Companies.ToList(), company,
                    _context.Outputs.GroupBy(item => item.Year).Select(item => new Filter-
ViewModel.Year
                    {
                        Name = item.Key.ToString(),
                        Number = item.Key
                    }).OrderBy(item => item.Number).ToList(), year),
                Items = itemList
            };
            return View(viewModel);
        }
    }
}
```

HomeController.cs:

```
namespace CourseWorkDb.Controllers
{
    [AllowAnonymous]
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            if (User.Identity.IsAuthenticated)
            {
                return View();
            }
            else
            {
                return RedirectToRoute(new { controller = "Account", action = "Login" });
            }
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpCon-
text.TraceIdentifier });
        }
    }
}
```

OutputsController.cs:

```
namespace CourseWorkDb.Controllers
{
    [Authorize]
    public class OutputsController : Controller
    {
        private const int PageSize = 10;
        private const string SessionKey1 = "ncr_company_filter";
        private const string SessionKey2 = "ncr_year_filter";

        private readonly RationingDbContext _context;
```

```
        public OutputsController(RationingDbContext context)
        {
            _context = context;
        }

        // GET: Outputs
        public async Task<IActionResult> Index(int? company, int? year, int page = 1)
        {
            if (company == null)
            {
                company = HttpContext.Session.GetInt32(SessionKey1);
            }

            if (year == null)
            {
                year = HttpContext.Session.GetInt32(SessionKey2);
            }

            IQueryable<Output> list = _context.Outputs.Include(n => n.Company).Include(n =>
n.Product);

            if (company != null)
            {
                if (company != 0)
                {
                    HttpContext.Session.SetInt32(SessionKey1, company.Value);
                    list = list.Where(p => p.CompanyId == company);
                }
            }

            if (year != null)
            {
                if (year != 0)
                {
                    HttpContext.Session.SetInt32(SessionKey2, year.Value);
                    list = list.Where(p => p.Year == year);
                }
            }

            int count = list.Count();
            var itemList = await list.Skip((page - 1) * PageSize).Take(PageSize).ToListA-
sync();

            IndexViewModel<Output> viewModel = new IndexViewModel<Output>
            {
                PageViewModel = new PageViewModel(count, page, PageSize),
                FilterViewModel = new FilterViewModel(_context.Companies.ToList(), company,
                    _context.Outputs.GroupBy(item => item.Year).Select(item => new Filter-
ViewModel.Year
                    {
                        Name = item.Key.ToString(),
                        Number = item.Key
                    }).OrderBy(item => item.Number).ToList(), year),
                Items = itemList
            };
            return View(viewModel);
        }

        // GET: Outputs/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
```

```
                return NotFound();
            }

            var output = await _context.Outputs
                .Include(n => n.Company)
                .Include(n => n.Product)
                .FirstOrDefaultAsync(m => m.OutputId == id);
            if (output == null)
            {
                return NotFound();
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", out-
put.CompanyId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
output.ProductId);
            return View(output);
        }

        public IActionResult Create()
        {
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name");
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit");
            return View();
        }

        // POST: Output/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,OutputPlan,OutputFact,Quar-
ter,Year,CompanyID,ProductId")] Output output)
        {
            if (_context.Outputs.FirstOrDefault(item => item.Year.Equals(output.Year)
                && item.Quarter.Equals(output.Quarter)
                && item.CompanyId.Equals(output.CompanyId)
                && item.ProductId.Equals(output.ProductId)) != null)
            {
                ModelState.AddModelError("Year", "Запись с таким годом и кварталом для дан-
ной продукции и организации уже существует!");
                ModelState.AddModelError("Quarter", "Запись с таким годом и кварталом для
данной продукции и организации уже существует!");
            }

            if (ModelState.IsValid)
            {
                _context.Add(output);
                _context.Add(new Output
                {
                    OutputPlan = 0,
                    OutputFact = 0,
                    Quarter = output.Quarter,
                    Year = output.Year,
                    CompanyId = output.CompanyId,
                    ProductId = output.ProductId,
                    Company = output.Company,
                    Product = output.Product
                });
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", out-
put.CompanyId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
output.ProductId);
            return View(output);
```

```
        }

        // POST: Output/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,OutputPlan,OutputFact,Quar-
ter,Year,CompanyId,ProductId")] Output output)
        {
            if (id != output.OutputId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(output);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!OutputExists(output.OutputId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", out-
put.CompanyId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
output.ProductId);
            return View(output);
        }

        // POST: Output/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var output = await _context.Outputs.FindAsync(id);
            _context.Outputs.Remove(output);
            var release = await _context.Releases.FindAsync(id);
            _context.Releases.Remove(release);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool OutputExists(int id)
        {
            return _context.Outputs.Any(e => e.OutputId == id);
        }
    }
}
```

ProductsController.cs:

```
namespace CourseWorkDb.Controllers
{
```

```
    [Authorize]
    public class ProductsController : Controller
    {
        private const int PageSize = 20;
        private readonly RationingDbContext _context;

        public ProductsController(RationingDbContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index(int page = 1)
        {
            IQueryable<Product> source = _context.Products;
            List<Product> list = await source.Skip((page - 1) *
PageSize).Take(PageSize).ToListAsync();

            IndexViewModel<Product> viewModel = new IndexViewModel<Product>
            {
                PageViewModel = new PageViewModel(await _context.Products.CountAsync(),
page, PageSize),
                Items = list
            };
            return View(viewModel);
        }

        // GET: Products/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var product = await _context.Products
                .FirstOrDefaultAsync(m => m.ProductId == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }

        // GET: Products/Create
        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,ProductName,MeasureUnit, Features,
Photo")] Product product)
        {
            if (ModelState.IsValid)
            {
                _context.Add(product);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(product);
        }
```

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,ProductName,MeasureUnit,Fea-
tures,Photo")] Product product)
        {
            if (id != product.ProductId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(product);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ProductExists(product.ProductId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(product);
        }

        // POST: Products/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var product = await _context.Products.FindAsync(id);
            _context.Products.Remove(product);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool ProductExists(int id)
        {
            return _context.Products.Any(e => e.ProductId == id);
        }
    }
}
```

ProductTypesController.cs

```
namespace CourseWorkDb.Controllers
{
    public class ProductTypesController : Controller
    {
        private readonly RationingDbContext _context;

        public ProductTypesController(RationingDbContext context)
        {
            _context = context;
        }
```

```
        // GET: ProductTypes
        public async Task<IActionResult> Index()
        {
            var rationingDbContext = _context.ProductTypes.Include(p => p.Product);
            return View(await rationingDbContext.ToListAsync());
        }

        // GET: ProductTypes/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var productType = await _context.ProductTypes
                .Include(p => p.Product)
                .FirstOrDefaultAsync(m => m.ProductTypeId == id);
            if (productType == null)
            {
                return NotFound();
            }

            return View(productType);
        }

        // GET: ProductTypes/Create
        public IActionResult Create()
        {
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Features");
            return View();
        }

        // POST: ProductTypes/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,ProductionType,ProductId")]
ProductType productType)
        {
            if (ModelState.IsValid)
            {
                _context.Add(productType);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Features",
productType.ProductId);
            return View(productType);
        }

        // GET: ProductTypes/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var productType = await _context.ProductTypes.FindAsync(id);
            if (productType == null)
            {
```

```
                return NotFound();
            }
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Features",
productType.ProductId);
            return View(productType);
        }

        // POST: ProductTypes/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,ProductionType,ProductId")]
ProductType productType)
        {
            if (id != productType.ProductTypeId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(productType);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ProductTypeExists(productType.ProductTypeId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Features",
productType.ProductId);
            return View(productType);
        }

        // GET: ProductTypes/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var productType = await _context.ProductTypes
                .Include(p => p.Product)
                .FirstOrDefaultAsync(m => m.ProductTypeId == id);
            if (productType == null)
            {
                return NotFound();
            }

            return View(productType);
        }
```

```
        // POST: ProductTypes/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var productType = await _context.ProductTypes.FindAsync(id);
            _context.ProductTypes.Remove(productType);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool ProductTypeExists(int id)
        {
            return _context.ProductTypes.Any(e => e.ProductTypeId == id);
        }
    }
}
```

ReleasesController.cs:

```
namespace CourseWorkDb.Controllers
{
    [Authorize]
    public class ReleasesController : Controller
    {
        private const int PageSize = 10;
        private const string SessionKey1 = "pad_company_filter";
        private const string SessionKey2 = "pad_year_filter";

        private readonly RationingDbContext _context;

        public ReleasesController(RationingDbContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index(int? company, int? year, int page = 1)
        {
            if (company == null)
            {
                company = HttpContext.Session.GetInt32(SessionKey1);
            }

            if (year == null)
            {
                year = HttpContext.Session.GetInt32(SessionKey2);
            }

            IQueryable<Release> list = _context.Releases.Include(p => p.Company).Include(p
=> p.Product);

            if (company != null)
            {
                HttpContext.Session.SetInt32(SessionKey1, company.Value);
                if (company != 0)
                {
                    list = list.Where(p => p.CompanyId == company);
                }
            }

            if (year != null)
            {
                HttpContext.Session.SetInt32(SessionKey2, year.Value);
```

```
                if (year != 0)
                {
                    list = list.Where(p => p.Year == year);
                }
            }

            int count = list.Count();
            var itemList = await list.Skip((page - 1) * PageSize).Take(PageSize).ToListA-
sync();

            IndexViewModel<Release> viewModel = new IndexViewModel<Release>
            {
                PageViewModel = new PageViewModel(count, page, PageSize),
                FilterViewModel = new FilterViewModel(_context.Companies.ToList(), company,
                    _context.Releases.GroupBy(item => item.Year).Select(item => new Filter-
ViewModel.Year
                    {
                        Name = item.Key.ToString(),
                        Number = item.Key
                    }).OrderBy(item => item.Number).ToList(), year),
                Items = itemList
            };
            return View(viewModel);
        }

        // GET: Releases/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var release = await _context.Releases
                .Include(p => p.Company)
                .Include(p => p.Product)
                .FirstOrDefaultAsync(m => m.ReleaseId == id);
            if (release == null)
            {
                return NotFound();
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", re-
lease.CompanyId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
release.ProductId);
            return View(release);
        }

        // GET: Releases/Create
        public IActionResult Create()
        {
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name");
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit");
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,ReleasePlan,ReleaseFact,Quar-
ter,Year,CompanyId,ProductId")] Release release)
        {
            if (_context.Releases.FirstOrDefault(item => item.Year.Equals(release.Year)
                && item.Quarter.Equals(release.Quarter)
                && item.CompanyId.Equals(release.CompanyId)
```

```
                && item.ProductId.Equals(release.ProductId)) != null)
            {
                ModelState.AddModelError("Year", "Запись с таким годом и кварталом для дан-
ной продукции и организации уже существует!");
                ModelState.AddModelError("Quarter", "Запись с таким годом и кварталом для
данной продукции и организации уже существует!");
            }

            if (ModelState.IsValid)
            {
                _context.Add(release);
                _context.Add(new Release
                {
                    ReleasePlan = 0,
                    ReleaseFact = 0,
                    Quarter = release.Quarter,
                    Year = release.Year,
                    CompanyId = release.CompanyId,
                    ProductId = release.ProductId,
                    Company = release.Company,
                    Product = release.Product
                });
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", re-
lease.CompanyId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
release.ProductId);
            return View(release);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,ReleasePlan,Release-
Fact,Quarter,Year,CompanyId,ProductId")] Release release)
        {
            if (id != release.ReleaseId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(release);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ReleaseExists(release.ReleaseId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["CompanyId"] = new SelectList(_context.Companies, "Id", "Name", re-
lease.CompanyId);
```

```
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "MeasureUnit",
release.ProductId);
            return View(release);
        }

        // POST: Releases/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var release = await _context.Releases.FindAsync(id);
            _context.Releases.Remove(release);
            var output = await _context.Outputs.FindAsync(id);
            _context.Outputs.Remove(output);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool ReleaseExists(int id)
        {
            return _context.Releases.Any(e => e.ReleaseId == id);
        }
    }
}
```

RolesController.cs:

```
namespace CourseWorkDb.Controllers
{
    [Authorize(Roles = Roles.Admin)]
    public class RolesController : Controller
    {
        private readonly RoleManager<IdentityRole> _roleManager;
        private readonly UserManager<User> _userManager;

        public RolesController(RoleManager<IdentityRole> roleManager, UserManager<User> us-
erManager)
        {
            _roleManager = roleManager;
            _userManager = userManager;
        }

        //смена ролей через админа
        public async Task<IActionResult> Edit(string userId)
        {
            User user = await _userManager.FindByIdAsync(userId);
            if (user != null)
            {
                IList<string> userRoles = await _userManager.GetRolesAsync(user);
                ChangeRoleViewModel model = new ChangeRoleViewModel
                {
                    UserId = user.Id,
                    UserEmail = user.Email,
                    AdminRole = Roles.Admin,
                    UserAdminRole = userRoles.FirstOrDefault(item => item.Equals(Roles.Ad-
min)) == null ? "" : Roles.Admin
                };
                return View(model);
            }

            return NotFound();
        }
        [HttpPost]

        //редактирование ролей
```

```
        public async Task<IActionResult> Edit(string userId, string adminRole)
        {
            User user = await _userManager.FindByIdAsync(userId);
            if (user != null)
            {
                if (adminRole == null)
                {
                    await _userManager.RemoveFromRoleAsync(user, Roles.Admin);
                }
                else
                {
                    await _userManager.AddToRoleAsync(user, Roles.Admin);
                }
                return RedirectToAction("Index", "Users");
            }

            return NotFound();
        }
    }
}
```

UsersControlle.cs:

```
namespace CourseWorkDb.Controllers
{
    [Authorize(Roles = Roles.Admin)]
    public class UsersController : Controller
    {
        private readonly UserManager<User> _userManager;

        public UsersController(UserManager<User> userManager)
        {
            _userManager = userManager;
        }

        public IActionResult Index() => View(_userManager.Users.ToList());

        public IActionResult Create() => View();

        [HttpPost]
        public async Task<IActionResult> Create(CreateUserViewModel model)
        {
            if (ModelState.IsValid)
            {
                User user = new User
                {
                    Email = model.Email,
                    UserName = model.Email,
                    Surname = model.Surname,
                    Name = model.Name,
                    MiddleName = model.MiddleName,
                    PhoneNumber = model.PhoneNumber
                };
                IdentityResult result = await _userManager.CreateAsync(user, model.Pass-
word);
                if (result.Succeeded)
                {
                    await _userManager.AddToRoleAsync(user, Roles.User);
                    return RedirectToAction("Index");
                }
                else
                {
                    foreach (IdentityError error in result.Errors)
                    {
                        ModelState.AddModelError(string.Empty, error.Description);
```

```
            }
        }
    }
    return View(model);
}

public async Task<IActionResult> Edit(string id)
{
    User user = await _userManager.FindByIdAsync(id);
    if (user == null)
    {
        return NotFound();
    }
    EditUserViewModel model = new EditUserViewModel
    {
        Id = user.Id,
        Email = user.Email,
        Surname = user.Surname,
        Name = user.Name,
        MiddleName = user.MiddleName,
        PhoneNumber = user.PhoneNumber
    };
    return View(model);
}

[HttpPost]
public async Task<IActionResult> Edit(EditUserViewModel model)
{
    if (ModelState.IsValid)
    {
        User user = await _userManager.FindByIdAsync(model.Id);
        if (user != null)
        {
            user.Email = model.Email;
            user.UserName = model.Email;
            user.Surname = model.Surname;
            user.Name = model.Name;
            user.MiddleName = model.MiddleName;
            user.PhoneNumber = model.PhoneNumber;

            IdentityResult result = await _userManager.UpdateAsync(user);
            if (result.Succeeded)
            {
                return RedirectToAction("Index");
            }
            else
            {
                foreach (IdentityError error in result.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }
        }
    }
    return View(model);
}

[HttpPost]
public async Task<ActionResult> Delete(string id)
{
    User user = await _userManager.FindByIdAsync(id);
    if (user != null && !user.UserName.Equals(User.Identity.Name))
    {
        await _userManager.DeleteAsync(user);
```

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

        public async Task<IActionResult> ChangePassword(string id)
        {
            User user = await _userManager.FindByIdAsync(id);
            if (user == null)
            {
                return NotFound();
            }
            ChangePasswordViewModel model = new ChangePasswordViewModel { Id = user.Id,
Email = user.Email };
            return View(model);
        }

        [HttpPost]
        public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                User user = await _userManager.FindByIdAsync(model.Id);
                if (user != null)
                {
                    IPasswordValidator<User> _passwordValidator = HttpContext.Request-
Services.GetService(typeof(IPasswordValidator<User>)) as IPasswordValidator<User>;
                    IPasswordHasher<User> _passwordHasher = HttpContext.RequestServices.Get-
Service(typeof(IPasswordHasher<User>)) as IPasswordHasher<User>;

                    IdentityResult result = await _passwordValidator.ValidateAsync(_userMan-
ager, user, model.NewPassword);
                    if (result.Succeeded)
                    {
                        user.PasswordHash = _passwordHasher.HashPassword(user, model.New-
Password);
                        await _userManager.UpdateAsync(user);
                        return RedirectToAction("Index");
                    }
                    else
                    {
                        foreach (IdentityError error in result.Errors)
                        {
                            ModelState.AddModelError(string.Empty, error.Description);
                        }
                    }
                }
                else
                {
                    ModelState.AddModelError(string.Empty, "Пользователь не найден");
                }
            }
            return View(model);
        }
    }
}
```

DbInitialozerMiddleware.cs:

```
namespace CourseWorkDb.Middleware
{
    public class DbInitializerMiddleware
    {
        private readonly RequestDelegate _next;

        public DbInitializerMiddleware(RequestDelegate next)
```

```
        {
            //инициализация базы данных
            _next = next;
        }

        public Task Invoke(HttpContext httpContext, RationingDbContext db)
        {
            DbInitializer.Initialize(db);
            return _next(httpContext);
        }
    }

    public static class DbMiddlewareExtensions
    {
        public static IApplicationBuilder UseInitializerMiddleware(this IApplicationBuilder
builder)
        {
            return builder.UseMiddleware<DbInitializerMiddleware>();
        }
    }
}
```

IdentityContext.cs:

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkDb.Models.Authentication
{
    public class IdentityContext : IdentityDbContext<User>
    {
        public IdentityContext(DbContextOptions<IdentityContext> options)
            : base(options)
        {
            Database.EnsureCreated();
        }
    }
}
```

RoleInitializer.cs:

```
using Microsoft.AspNetCore.Identity;
using System.Threading.Tasks;

namespace CourseWorkDb.Models.Authentication
{
    public class RoleInitializer
    {
        public static async Task InitializeAsync(UserManager<User> userManager,
            RoleManager<IdentityRole> roleManager)
        {
            string adminEmail = "admin@gmail.com";
            string adminSurname = "Сетко";
            string adminName = "Анастасия";
            string adminMiddleName = "Игоревна";
            string adminPhoneNumber = "375445487111";
            string adminPassword = "_Aa123456";

            if (await roleManager.FindByNameAsync(Roles.Admin) == null)
            {
                await roleManager.CreateAsync(new IdentityRole(Roles.Admin));
            }
            if (await roleManager.FindByNameAsync(Roles.User) == null)
            {
                await roleManager.CreateAsync(new IdentityRole(Roles.User));
```

```
            }

            if (await userManager.FindByNameAsync(adminEmail) == null)
            {
                User admin = new User
                {
                    Email = adminEmail,
                    UserName = adminEmail,
                    PhoneNumber = adminPhoneNumber,
                    Surname = adminSurname,
                    Name = adminName,
                    MiddleName = adminMiddleName
                };
                IdentityResult result = await userManager.CreateAsync(admin, adminPassword);
                if (result.Succeeded)
                {
                    await userManager.AddToRolesAsync(admin, new string[]{ Roles.User,
Roles.Admin});
                }
            }
        }
    }
}
```

Role.cs:

```
namespace CourseWorkDb.Models.Authentication
{
    public static class Roles
    {
        public const string User = "user";
        public const string Admin = "admin";

        public const string UserOrAdmin = "user, admin";
    }
}
```

RussianIdentityErrorDescriber.cs

```
using Microsoft.AspNetCore.Identity;

namespace CourseWorkDb.Models.Authentication
{
    public class RussianIdentityErrorDescriber : IdentityErrorDescriber
    {
        public override IdentityError DuplicateUserName(string userName)
        {
            return new IdentityError
            {
                Code = nameof(DuplicateUserName),
                Description = $"Email '{userName}' уже занят другим пользователем"
            };
        }

        public override IdentityError PasswordRequiresNonAlphanumeric()
        {
            return new IdentityError
            {
                Code = nameof(PasswordRequiresNonAlphanumeric),
                Description = "Пароль должен содержать как минимум один не буквенно-числен-
ный символ"
            };
        }

        public override IdentityError PasswordRequiresDigit()
```

```
        {
            return new IdentityError
            {
                Code = nameof(PasswordRequiresDigit),
                Description = "Пароль должен содержать как минимум одну цифру"
            };
        }

        public override IdentityError PasswordRequiresLower()
        {
            return new IdentityError
            {
                Code = nameof(PasswordRequiresLower),
                Description = "Пароль должен содержать как минимум одну строчную букву"
            };
        }

        public override IdentityError PasswordRequiresUpper()
        {
            return new IdentityError
            {
                Code = nameof(PasswordRequiresUpper),
                Description = "Пароль должен содержать как минимум одну прописную букву"
            };
        }
    }
}
```

User.cs

```
using Microsoft.AspNetCore.Identity;

namespace CourseWorkDb.Models.Authentication
{
    public class User : IdentityUser
    {
        public string Surname { get; set; }
        public string Name { get; set; }
        public string MiddleName { get; set; }
        public override string Email { get; set; }
    }
}
```

Company.cs:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.Models.Tables
{
    public partial class Company
    {
        public Company()
        {
            Output = new HashSet<Output>();
            Release = new HashSet<Release>();
        }

        public int CompanyId { get; set; }

        [Display(Name = "Название предприятия")]
        [Required(ErrorMessage = "Укажите название предприятия")]
        [StringLength(40, MinimumLength = 5, ErrorMessage = "Минимальная длина названия ор-
ганизации должна быть 5 символов")]
        public string CompanyName { get; set; }
```

```
        [Display(Name = "Форма собственности")]
        [Required(ErrorMessage = "Укажите форму собственности")]
        [StringLength(30, MinimumLength = 3, ErrorMessage = "Минимальная длина формы соб-
ственности должна быть 3 символа")]
        public string FormOfOwnership { get; set; }

        [Display(Name = "Вид деятельности")]
        [Required(ErrorMessage = "Укажите вид деятелности")]
        [StringLength(30, MinimumLength = 3, ErrorMessage = "Минимальная длина вида деятель-
ности 3 символа")]
        public string ActivityType { get; set; }

        [Display(Name = "ФИО директора")]
        [Required(ErrorMessage = "Укажите ФИО директора")]
        [StringLength(50, MinimumLength = 5, ErrorMessage = "Минимальная длина ФИО директора
должна быть 5 символов")]
        public string HeadName { get; set; }


        public virtual ICollection<Output> Output { get; set; }
        public virtual ICollection<Release> Release { get; set; }
    }
}
```

Output.cs:

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.Models.Tables
{
    public partial class Output
    {
        public int OutputId { get; set; }

        [Display(Name = "Планируемый выпуск продукции")]
        [Required(ErrorMessage = "Укажите планируемый выпуск продукции")]
        public int OutputPlan { get; set; }

        [Display(Name = "Фактический выпуск продукции")]
        [Required(ErrorMessage = "Укажите фактический выпуск продукции")]
        public int OutputFact { get; set; }

        [Display(Name = "Квартал")]
        [Required(ErrorMessage = "Укажите квартал")]
        [Range(1, 4, ErrorMessage = "Квартал должен быть равен 1, 2, 3 или 4")]
        public short Quarter { get; set; }

        [Display(Name = "Год")]
        [Required(ErrorMessage = "Укажите год")]
        [Range(1970, 2050, ErrorMessage = "Год должен быть равен не менее 1970 и не более
2050")]
        public short Year { get; set; }

        public int CompanyId { get; set; }
        public int ProductId { get; set; }

        public virtual Company Company { get; set; }
        public virtual Product Product { get; set; }
    }
}
```

Product.cs:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
```

```
namespace CourseWorkDb.Models.Tables
{
    public partial class Product
    {
        public Product()
        {
            Output = new HashSet<Output>();
            Release = new HashSet<Release>();
            ProductType = new HashSet<ProductType>();
        }

        [Display(Name = "Номер")]
        public int ProductId { get; set; }

        [Display(Name = "Наименование продукции")]
        [Required(ErrorMessage = "Укажите наименование продукции")]
        [StringLength(50, ErrorMessage = "Максимальная длина наименования продукции должна
быть 50 символов")]
        public string ProductName{ get; set; }

        [Display(Name = "Единица измерения")]
        [Required(ErrorMessage = "Укажите единицу измерения")]
        [StringLength(50, ErrorMessage = "Максимальная длина единицы измерения должна быть
50 символов")]
        public string MeasureUnit { get; set; }

        [Display(Name = "Описание продукта")]
        [Required(ErrorMessage = "Укажите описание продукта")]
        [StringLength(50, ErrorMessage = "Максимальная длина описания продукта должна быть
50 символов")]
        public string Features { get; set; }

        [Display(Name = "Фото продукта")]
        public byte[] Photo { get; set; }


        public virtual ICollection<Output> Output { get; set; }
        public virtual ICollection<Release> Release { get; set; }
        public virtual ICollection<ProductType> ProductType { get; set; }
    }
}
```

ProductType.cs

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.Models.Tables
{
    public partial class ProductType
    {


        [Display(Name = "Номер")]
        public int ProductTypeId { get; set; }

        [Display(Name = "Вид продукции")]
        [Required(ErrorMessage = "Укажите вид продукции")]
        [StringLength(50, ErrorMessage = "Максимальная длина вида продукции должна быть 50
символов")]
        public string ProductionType { get; set; }

        public int ProductId { get; set; }

        public virtual Product Product { get; set; }
    }
}
```

Release.cs

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.Models.Tables
{
    public partial class Release
    {
        public int ReleaseId { get; set; }

        [Display(Name="Планипуемая релизация продукции")]
        [Required(ErrorMessage = "Укажите планируемую реализацию продукции")]
        public int ReleasePlan { get; set; }

        [Display(Name = "Фактическая реализация продукции")]
        [Required(ErrorMessage = "Укажите фактическую реализацию продукции")]
        public int ReleaseFact { get; set; }

        [Display(Name = "Квартал")]
        [Required(ErrorMessage = "Укажите квартал")]
        [Range(1, 4, ErrorMessage = "Квартал должен быть равен 1, 2, 3 или 4")]
        public short Quarter { get; set; }

        [Display(Name = "Год")]
        [Required(ErrorMessage = "Укажите год")]
        [Range(1970, 2050, ErrorMessage = "Год должен быть равен не менее 1970 и не более
2050")]
        public short Year { get; set; }

        public int CompanyId { get; set; }
        public int ProductId { get; set; }

        public virtual Company Company { get; set; }
        public virtual Product Product { get; set; }
    }
}
```

DbInitializer

```
using CourseWorkDb.Models.Tables;
using System;
using System.Linq;
using System.Text;

namespace CourseWorkDb.Models
{
    public static class DbInitializer
    {
        public static void Initialize(RationingDbContext db)
        {
            db.Database.EnsureCreated();

            if (db.Products.Any()) return;

            int onSideOne = 100;
            int onSideMany = 1000;

            Random random = new Random();

            Company con = new Company();

            con.FormOfOwnership = CreateRandomString(random);

            for (int i = 0; i < onSideOne; i++)
            {
                db.Products.Add(new Product
```

```
            {
                ProductName = CreateRandomString(random),
                MeasureUnit = CreateRandomString(random),
                Features = CreateRandomString(random),
                Photo = null
            });
        }

        for (int i = 0; i < onSideOne; i++)
        {
            db.ProductTypes.Add(new ProductType
            {
                ProductionType = CreateRandomString(random),

            });
        }

        db.SaveChanges();
        for (int i = 0; i < onSideOne; i++)
        {
            db.Companies.Add(new Company
            {
                CompanyName = CreateRandomString(random),
                FormOfOwnership = CreateRandomString(random),
                HeadName = CreateRandomString(random),
                ActivityType = CreateRandomString(random),
            });
        }
        db.SaveChanges();

        for (int i = 0; i < onSideMany; i++)
        {
            int quarter = random.Next(1, 5);
            int year = random.Next(1970, 2050);
            int companyId = i % onSideOne + 1;
            int productId = i % onSideOne + 1;

            db.Outputs.Add(new Output
            {
                OutputPlan = random.Next(1, 1000),
                OutputFact = (int)(random.NextDouble() * random.Next(100, 1000000)),
                Quarter = (short)quarter,
                Year = (short)year,
                CompanyId = companyId,
                ProductId = productId
            });
            db.Releases.Add(new Release
            {
                ReleasePlan = random.Next(1, 1000),
                ReleaseFact = (int)(random.NextDouble() * random.Next(100, 1000000)),
                Quarter = (short)quarter,
                Year = (short)year,
                CompanyId = companyId,
                ProductId = productId,

            });
        }

        db.SaveChanges();
    }

    private static string CreateRandomString(Random random)
    {
```

```
            string letters = "ABCDEFGHIJKLMNOPQRSTU-
VWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзиёклмнопрстуфхцчшщъыьэюяabcdefghijklmnopqrstu
vwxyz";

            StringBuilder builder = new StringBuilder();
            for (int i = 5; i <= 30; i++)
            {
                builder.Append(letters[random.Next(0, letters.Length - 1)]);
            }
            return builder.ToString();
        }

    }
}
```

ErrorViewModel.cs:

```
namespace CourseWorkDb.Models
{
    public class ErrorViewModel
    {
        public string RequestId { get; set; }

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

RationingDbContect:

```
using CourseWorkDb.Models.Tables;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkDb.Models
{
    public partial class RationingDbContext : DbContext
    {
        public RationingDbContext(DbContextOptions<RationingDbContext> options)
            : base(options)
        {

        }

        public virtual DbSet<Output> Outputs { get; set; }
        public virtual DbSet<Company> Companies { get; set; }
        public virtual DbSet<Release> Releases { get; set; }
        public virtual DbSet<Product> Products { get; set; }
        public virtual DbSet<ProductType> ProductTypes { get; set; }
    }
}
```

ChangePAsswordViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.ViewModels
{
    public class ChangePasswordViewModel
    {
        public string Id { get; set; }
        public string Email { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должен содержать минимум {2} и максимум {1}
символов", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
```

```
        public string NewPassword { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [Compare("NewPassword", ErrorMessage = "Пароли не совпадают")]
        [DataType(DataType.Password)]
        [Display(Name = "Подтвердить пароль")]
        public string NewPasswordConfirm { get; set; }
    }
}
```

ChangeRole new model.cs:

```
using Microsoft.AspNetCore.Identity;

namespace CourseWorkDb.ViewModels
{
    public class ChangeRoleViewModel
    {
        public string UserId { get; set; }
        public string UserEmail { get; set; }
        public string AdminRole { get; set; }
        public string UserAdminRole { get; set; }
    }
}
```

CreateUserViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.ViewModels
{
    public class CreateUserViewModel
    {
        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [EmailAddress(ErrorMessage = "Текст в поле {0} не является правильной записью
email")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Пользователь с таким {0} уже суще-
ствует")]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должна содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[A-ЯЁA-Z]{1}[a-яA-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должна
начинаться с большой буквы и содержать только буквы")]
        [DataType(DataType.Text)]
        [Display(Name = "Фамилия")]
        public string Surname { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должно содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[A-ЯЁA-Z]{1}[a-яA-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должно
начинаться с большой буквы и содержать только буквы")]
        [DataType(DataType.Text)]
        [Display(Name = "Имя")]
        public string Name { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должно содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[A-ЯЁA-Z]{1}[a-яA-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должно
начинаться с большой буквы и содержать только буквы")]
        [DataType(DataType.Text)]
        [Display(Name = "Отчество")]
        public string MiddleName { get; set; }
```

```
        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [Phone(ErrorMessage = "Текст в поле {0} не является правильной записью номера теле-
фона")]
        [DataType(DataType.PhoneNumber)]
        [Display(Name = "Номер телефона")]
        public string PhoneNumber { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должен содержать минимум {2} и максимум {1}
символов", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [Compare("Password", ErrorMessage = "Пароли не совпадают")]
        [DataType(DataType.Password)]
        [Display(Name = "Подтвердить пароль")]
        public string PasswordConfirm { get; set; }
    }
}
```

EditUserViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.ViewModels
{
    public class EditUserViewModel
    {
        public string Id { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [EmailAddress(ErrorMessage = "Текст в поле {0} не является правильной записью
email")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Пользователь с таким {0} уже суще-
ствует")]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должна содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[А-ЯЁA-Z]{1}[а-яА-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должна
начинаться с большой буквы и содержать только буквы")]
        [DataType(DataType.Text)]
        [Display(Name = "Фамилия")]
        public string Surname { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должно содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[А-ЯЁA-Z]{1}[а-яА-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должно
начинаться с большой буквы и содержать только буквы")]
        [DataType(DataType.Text)]
        [Display(Name = "Имя")]
        public string Name { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [StringLength(30, ErrorMessage = "{0} должно содержать минимум {2} и максимум {1}
символов", MinimumLength = 2)]
        [RegularExpression("[А-ЯЁA-Z]{1}[а-яА-ЯёЁa-zA-Z-]{1,30}", ErrorMessage = "{0} должно
начинаться с большой буквы и содержать только буквы")]
```

```
        [DataType(DataType.Text)]
        [Display(Name = "Отчество")]
        public string MiddleName { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [Phone(ErrorMessage = "Текст в поле {0} не является правильной записью номера теле-
фона")]
        [DataType(DataType.PhoneNumber)]
        [Display(Name = "Номер телефона")]
        public string PhoneNumber { get; set; }
    }
}
```

FilterViewModel.cs:

```
using CourseWorkDb.Models.Tables;
using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;

namespace CourseWorkDb.ViewModels
{
    public class FilterViewModel
    {
        public SelectList Companies { get; private set; }
        public int? SelectedCompany { get; private set; }
        public SelectList Years { get; private set; }
        public int? SelectedYear { get; private set; }

        public FilterViewModel(List<Company> companies, int? company, List<Year> years, int?
year)
        {
            companies.Insert(0, new Company { CompanyName = "Все", CompanyId = 0 });
            Companies = new SelectList(companies, "Id", "Name", company);
            SelectedCompany = company;

            years.Insert(0, new Year { Name = "Все", Number = 0 });
            Years = new SelectList(years, "Number", "Name", year);
            SelectedYear = year;
        }


        public class Year
        {
            public int Number { get; set; }
            public string Name { get; set; }
        }
    }
}
```

IndexViewModel.cs:

```
using System.Collections.Generic;

namespace CourseWorkDb.ViewModels
{
    public class IndexViewModel<T>
    {
        public IEnumerable<T> Items { get; set; }
        public PageViewModel PageViewModel { get; set; }
        public FilterViewModel FilterViewModel { get; set; }
    }
}
```

LoginViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace CourseWorkDb.ViewModels
{
    public class LoginViewModel
    {
        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [EmailAddress]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required(ErrorMessage = "{0} является обязательным полем для заполнения")]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        public string ReturnUrl { get; set; }
    }
}
```

PageViewModel.cs:

```
using System;

namespace CourseWorkDb.ViewModels
{
    public class PageViewModel
    {
        public int PageNumber { get; private set; }
        public int TotalPages { get; private set; }

        public PageViewModel(int count, int pageNumber, int pageSize)
        {
            PageNumber = pageNumber;
            TotalPages = (int)Math.Ceiling(count / (double)pageSize);
        }

        public bool HasPreviousPage
        {
            get
            {
                return (PageNumber > 1);
            }
        }

        public bool HasSecondPreviousPage
        {
            get
            {
                return (PageNumber - 1 > 1);
            }
        }

        public bool HasNextPage
        {
            get
            {
                return (PageNumber < TotalPages);
            }
        }
```

```
        public bool HasSecondNextPage
        {
            get
            {
                return (PageNumber + 1 < TotalPages);
            }
        }
    }
}
```

Login.cshtml:

```
@model CourseWorkDb.ViewModels.LoginViewModel
@{
    ViewData["Title"] = "Авторизация";
}

<div class="container">
    <div class="row justify-content-md-center">
        <div class="col-md-auto">
            <form class="form-horizontal" method="post" asp-controller="Account" asp-ac-
tion="Login" asp-route-returnUrl="@Model.ReturnUrl">
                <span class="heading">Авторизация</span>
                <div asp-validation-summary="ModelOnly"></div>
                <div class="form-group">
                    <input type="email" class="form-control" asp-for="Email" place-
holder="Email" required>
                </div>
                <div class="form-group help">
                    <input type="password" class="form-control" asp-for="Password" place-
holder="Пароль" required>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-default">Войти</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

Comparison.cshtml:

```
@model IndexViewModel<ComparisonViewModel>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Анализ выполнения плана выпуска продукции";
}

<br />
<form asp-action="Comparison" method="get">
    <div class="form-row col-md-8">
        <div class="col-2">
            <label class="control-label">Название предприятия </label>
        </div>
        <div class="col-4">
            @Html.DropDownList("company", Model.FilterViewModel.Companies as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col-1">
            <label class="control-label">Год </label>
        </div>
```

```
        <div class="col-3">
            @Html.DropDownList("year", Model.FilterViewModel.Years as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col">
            <input type="submit" value="Найти" class="btn btn-secondary" />
        </div>
    </div>
</form>

<br />
<div class="table-responsive">
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col">Предприятие</th>
                <th scope="col">Единица измерения</th>
                <th scope="col">Фактический выпуск продукции</th>
                <th scope="col">План выпуска продукции</th>
                <th scope="col">Год</th>
                <th scope="col">Квартал</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
                <tr onclick="window.location.href='/Companies/Details/@item.CompanyId'; re-
turn false">
                    <td>
                        @Html.DisplayFor(modelItem => item.CompanyName)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.MeasureUnit)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.OutputFact)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.OutputPlan)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Year)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Quarter)
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

<page-link page-model="Model.PageViewModel" page-action="Comparison"
           page-url-organization="@(Model.FilterViewModel.SelectedCompany)"
           page-url-year="@(Model.FilterViewModel.SelectedYear)"></page-link>
```

Create.cshtml:

```
@model CourseWorkDb.Models.Tables.Company

@{
    ViewData["Title"] = "Добавление";
}
```

```
<h4>Предприятие</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="CompanyName" class="control-label"></label>
                <input asp-for="CompanyName" class="form-control" />
                <span asp-validation-for="CompanyName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="FormOfOwnership" class="control-label"></label>
                <input asp-for="FormOfOwnership" class="form-control" />
                <span asp-validation-for="FormOfOwnership" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ActivityType" class="control-label"></label>
                <input asp-for="ActivityType" class="form-control" />
                <span asp-validation-for="ActivityType" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="HeadName" class="control-label"></label>
                <input asp-for="HeadName" class="form-control" />
                <span asp-validation-for="HeadName" class="text-danger"></span>
            </div>

            <div class="form-group">
                <input type="submit" value="Добавить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Details.cshtml:

```
@model CourseWorkDb.Models.Tables.Company

@{
    ViewData["Title"] = "Предприятие";
}


<h5>Предприятие</h5>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" asp-action="Edit">
            <input type="hidden" asp-for="CompanyId" readonly />
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="CompanyName" class="control-label"></label>
                <input asp-for="CompanyName" class="form-control" />
                <span asp-validation-for="CompanyName" class="text-danger"></span>
            </div>
            <div class="form-group">
```

```
                <label asp-for="FormOfOwnership" class="control-label"></label>
                <input asp-for="FormOfOwnership" class="form-control" />
                <span asp-validation-for="FormOfOwnership" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ActivityType" class="control-label"></label>
                <input asp-for="ActivityType" class="form-control" />
                <span asp-validation-for="ActivityType" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="HeadName" class="control-label"></label>
                <input asp-for="HeadName" class="form-control" />
                <span asp-validation-for="HeadName" class="text-danger"></span>
            </div>

        </form>
        <form method="post" asp-action="Delete">
            <input type="hidden" asp-for="CompanyId" />
            <input type="submit" value="Удалить" class="btn btn-danger" />
        </form>
    </div>
</div>

<br />
<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index.cshtml:

```
@model IndexViewModel<Company>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Список предприятийй";
}

<br/>
<form asp-action="Index" method="get">
    <div class="form-row col-md-6">

        <div class="col">
            <input type="text" name="searchByName" value="@ViewData["CurrentSearch"]"
class="form-control" placeholder="Название предприятия" />
        </div>
        <div class="col">
            <input type="submit" value="Найти" class="btn btn-secondary" />
        </div>
    </div>
</form>

<br />
<div class="col">
    <a class="btn btn-info" asp-action="Index">Полный список</a>
</div>
<br/>

<div class="table-responsive">
    <table class="table table-hover">
```

```
        <thead>
            <tr>
                <th scope="col">Название предприятия</th>
                <th scope="col">Тип производственной деятельности</th>
                <th scope="col">ФИО Директора</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
                <tr onclick="window.location.href='/Companies/Details/@item.CompanyId'; re-
turn false">
                    <td>
                        @Html.DisplayFor(modelItem => item.CompanyName)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.ActivityType)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.HeadName)
                    </td>

                </tr>
            }
        </tbody>
    </table>
</div>

<page-link page-model="Model.PageViewModel" page-action="Index"></page-link>


Violators.cshtml:

@model IndexViewModel<ComparisonViewModel>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Нарушители норм потребления";
}

<br />
<form asp-action="Violators" method="get">
    <div class="form-row col-md-8">
        <div class="col-2">
            <label class="control-label">Название предприятия </label>
        </div>
        <div class="col-4">
            @Html.DropDownList("company", Model.FilterViewModel.Companies as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col-1">
            <label class="control-label">Год </label>
        </div>
        <div class="col-3">
            @Html.DropDownList("year", Model.FilterViewModel.Years as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col">
            <input type="submit" value="Найти" class="btn btn-secondary" />
        </div>
    </div>
</form>
```

```
<br />
<div class="table-responsive">
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col">Предприятие</th>
                <th scope="col">Единица измерения</th>
                <th scope="col">Превышение нормы потребления на единицу товара</th>
                <th scope="col">Год</th>
                <th scope="col">Квартал</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
                <tr onclick="window.location.href='/Companies/Details/@item.CompanyId'; re-
turn false">
                    <td>
                        @Html.DisplayFor(modelItem => item.CompanyName)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.MeasureUnit)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Excess)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Year)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Quarter)
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

<page-link page-model="Model.PageViewModel" page-action="Violators"
           page-url-organization="@(Model.FilterViewModel.SelectedCompany)"
           page-url-year="@(Model.FilterViewModel.SelectedYear)"></page-link>
```

Index.cshtml:

```
@{
    ViewData["Title"] = "CourseWork";
}

<br/>
<div style="align-content: center">
    <h4 style="text-align: center">
        Курсовая работа
    </h4>
    <h4 style="text-align: center">
        Приложение для учёта выпуска и реализации сходных видов продукции»
    </h4>
</div>
```

Create.cshtml:

```
@model CourseWorkDb.Models.Tables.Output

@{
```

```
    ViewData["Title"] = "Добавление";
}

<h4>Выпуск продукции</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="OutputPlan" class="control-label"></label>
                <input asp-for="OutputPlan" class="form-control" data-val-number="Пожалуйста
введите число." />
                <span asp-validation-for="OutputPlan" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OutputFact" class="control-label"></label>
                <input asp-for="OutputFact" class="form-control" />
                <span asp-validation-for="OutputFact" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quarter" class="control-label"></label>
                <input asp-for="Quarter" class="form-control" />
                <span asp-validation-for="Quarter" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CompanyId" class="control-label"></label>
                <select asp-for="CompanyId" class="form-control" asp-items="ViewBag.Compa-
nyId"></select>
            </div>
            <div class="form-group">
                <label asp-for="ProductId" class="control-label"></label>
                <select asp-for="ProductId" class="form-control" asp-items="View-
Bag.ProductId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Добавить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
    <script type="text/javascript">
        $.validator.methods.range = function (value, element, param) {
            var globalizedValue = value.replace(",", ".");
            return this.optional(element) || (globalizedValue >= param[0] && globalizedValue
<= param[1]);
        }
        $.validator.methods.number = function (value, element) {
            return this.optional(element) || /^-
?(?:\d+|\d{1,3}(?:[\s\.,]\d{3})+)(?:[\.,]\d+)?$/.test(value);
        }
    </script>
```

```
}
```

Details.cshtml:

```
@model CourseWorkDb.Models.Tables.Output

@{
    ViewData["Title"] = "Выпуск продукции";
}


    <h4>Выпуск продукции"</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="CompanyId" />
            <div class="form-group">
                <label asp-for="OutputPlan" class="control-label"></label>
                <input asp-for="OutputPlan" class="form-control" data-val-number="Пожалуйста
введите число." />
                <span asp-validation-for="OutputPlan" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OutputFact" class="control-label"></label>
                <input asp-for="OutputFact" class="form-control" />
                <span asp-validation-for="OutputFact" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quarter" class="control-label"></label>
                <input asp-for="Quarter" class="form-control" readonly/>
                <span asp-validation-for="Quarter" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" readonly/>
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CompanyId" class="control-label"></label>
                <select asp-for="CompanyId" class="form-control" asp-items="ViewBag.Compa-
nyId"></select>
                <span asp-validation-for="CompanyId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ProductId" class="control-label"></label>
                <select asp-for="ProductId" class="form-control" asp-items="View-
Bag.ProductId"></select>
                <span asp-validation-for="ProductId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Редактировать" class="btn btn-primary" />
            </div>
        </form>
        <form method="post" asp-action="Delete">
            <input type="hidden" asp-for="CompanyId" />
            <input type="submit" value="Удалить" class="btn btn-danger" />
        </form>
    </div>
</div>

<br />
<div>
```

```
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index.cshtml:

```
@model IndexViewModel<Output>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Список выпуска продукции";
}

<br />
<form asp-action="Index" method="get">
    <div class="form-row col-md-8">
        <div class="col-2">
            <label class="control-label">Название предприятия </label>
        </div>
        <div class="col-4">
            @Html.DropDownList("company", Model.FilterViewModel.Companies as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col-1">
            <label class="control-label">Год </label>
        </div>
        <div class="col-3">
            @Html.DropDownList("year", Model.FilterViewModel.Years as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col">
            <input type="submit" value="Найти" class="btn btn-secondary" />
        </div>
    </div>
</form>

<br />
<div class="table-responsive">
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col">Предприятие</th>
                <th scope="col">Фактический выпуск продукции</th>
                <th scope="col">План выпуска продукции</th>
                <th scope="col">Год</th>
                <th scope="col">Квартал</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
                <tr onclick="window.location.href='/Outputs/Details/@item.OutputId'; return
false">
                    <td>
                        @Html.DisplayFor(modelItem => item.Company.CompanyName)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.OutputPlan)
                    </td>
                    <td>
```

```
                    @Html.DisplayFor(modelItem => item.OutputFact)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Year)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Quarter)
                </td>
            </tr>
        }
    </tbody>
</table>
</div>

<page-link page-model="Model.PageViewModel" page-action="Violators"
           page-url-organization="@(Model.FilterViewModel.SelectedCompany)"
           page-url-year="@(Model.FilterViewModel.SelectedYear)"></page-link>
```

Create.cshtml:

```
@model CourseWorkDb.Models.Tables.Product

@{
    ViewData["Title"] = "Добавление";
}

<h4>Вид продукции</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="MeasureUnit" class="control-label"></label>
                <input asp-for="MeasureUnit" class="form-control" />
                <span asp-validation-for="MeasureUnit" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Features" class="control-label"></label>
                <input asp-for="Features" class="form-control" />
                <span asp-validation-for="Features" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Photo" class="control-label"></label>
                <input asp-for="Photo" class="form-control" />
                <span asp-validation-for="Photo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Добавить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Details.cshtml:

```
@model CourseWorkDb.Models.Tables.Product

@{
    ViewData["Title"] = "Продукт";
}

<h5>Вид продукции</h5>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <input type="hidden" asp-for="ProductId" readonly />
            <div class="form-group">
                <label asp-for="MeasureUnit" class="control-label"></label>
                <input asp-for="MeasureUnit" class="form-control" />
                <span asp-validation-for="MeasureUnit" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Features" class="control-label"></label>
                <input asp-for="Features" class="form-control" />
                <span asp-validation-for="Features" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Photo" class="control-label"></label>
                <input asp-for="Photo" class="form-control" />
                <span asp-validation-for="Photo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Редактировать" class="btn btn-primary" />
            </div>
        </form>
        <form method="post" asp-action="Delete">
            <input type="hidden" asp-for="ProductId" />
            <input type="submit" value="Удалить" class="btn btn-danger" />
        </form>
    </div>
</div>

<br/>
<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index.cshtml:

```
@model IndexViewModel<Product>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Список продукций";
    string content = "image/jpeg";
}

<div class="table-responsive">
    <table class="table table-hover">
        <thead>
            <tr>
```

```
                <th scope="col">Номер</th>
                <th scope="col">Наименование продукции</th>
                <th scope="col">Единица измерения</th>
                <th scope="col">Описание</th>
                <th scope="col">Фото</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
            <tr onclick="window.location.href='/Products/Details/@item.ProductId'; return
false">
                <td>
                    @Html.DisplayFor(modelItem => item.ProductId)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ProductName)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.MeasureUnit)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Features)
                </td>
                @*<td>
                    @Html.DisplayFor(modelItem => item.Photo)
                </td>*@
            <td>
    @*{string src = string.Format("data:{0};base64,{1}", content,

                        Convert.ToBase64String(item.Photo));
    }*@

                <img src=" @*src*@" />
            </td>
            </tr>
            }
        </tbody>
    </table>
</div>


<page-link page-model="Model.PageViewModel" page-action="Index"></page-link>

Create.cshtml:

@model CourseWorkDb.Models.Tables.ProductType

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>ProductType</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="ProductionType" class="control-label"></label>
                <input asp-for="ProductionType" class="form-control" />
                <span asp-validation-for="ProductionType" class="text-danger"></span>
```

```
            </div>

            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

Delete.cshtml:

```
@model CourseWorkDb.Models.Tables.ProductType

@{
    ViewData["Title"] = "Delete";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>ProductType</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.ProductionType)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ProductionType)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Product)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Product.Features)
        </dd class>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="ProductTypeId" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

Details.cshtml:

```
@model CourseWorkDb.Models.Tables.ProductType

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>ProductType</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
```

```
            @Html.DisplayNameFor(model => model.ProductionType)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ProductionType)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Product)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Product.Features)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.ProductTypeId">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

Edit.cshtml:

```
@model CourseWorkDb.Models.Tables.ProductType

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

<h4>ProductType</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="ProductTypeId" />
            <div class="form-group">
                <label asp-for="ProductionType" class="control-label"></label>
                <input asp-for="ProductionType" class="form-control" />
                <span asp-validation-for="ProductionType" class="text-danger"></span>
            </div>

            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

Index.cshtml:

```
@model IEnumerable<CourseWorkDb.Models.Tables.ProductType>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
```

```
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.ProductionType)
            </th>

            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.ProductionType)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.ProductTypeId">Edit</a> |
                <a asp-action="Details" asp-route-id="@item.ProductTypeId">Details</a> |
                <a asp-action="Delete" asp-route-id="@item.ProductTypeId">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```

Create.cshtml:

```
@model CourseWorkDb.Models.Tables.Release

@{
    ViewData["Title"] = "Добавление";
}

<h4>Реализация продукции</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="ReleasePlan" class="control-label"></label>
                <input asp-for="ReleasePlan" class="form-control" />
                <span asp-validation-for="ReleaseFact" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ReleaseFact" class="control-label"></label>
                <input asp-for="ReleaseFact" class="form-control" data-val-num-
ber="Пожалуйста введите число." />
                <span asp-validation-for="ReleaseFact" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quarter" class="control-label"></label>
                <input asp-for="Quarter" class="form-control" />
                <span asp-validation-for="Quarter" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CompanyId" class="control-label"></label>
```

```
                <select asp-for="CompanyId" class="form-control" asp-items="ViewBag.Compa-
nyId"></select>
            </div>
            <div class="form-group">
                <label asp-for="ProductId" class="control-label"></label>
                <select asp-for="ProductId" class="form-control" asp-items="View-
Bag.ProductId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Добавить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
    <script type="text/javascript">
        $.validator.methods.range = function (value, element, param) {
            var globalizedValue = value.replace(",", ".");
            return this.optional(element) || (globalizedValue >= param[0] && globalizedValue
<= param[1]);
        }
        $.validator.methods.number = function (value, element) {
            return this.optional(element) || /^-
?(?:\d+|\d{1,3}(?:[\s\.,]\d{3})+)(?:[\.,]\d+)?$/.test(value);
        }
    </script>
}
```

Details.cshtml:

```
@model CourseWorkDb.Models.Tables.Release

@{
    ViewData["Title"] = "Реализация продукции";
}

<h4>Производство и потребление</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="ReleaseId" />
            <div class="form-group">
                <label asp-for="ReleasePlan" class="control-label"></label>
                <input asp-for="ReleasePlan" class="form-control" data-val-num-
ber="Пожалуйста введите число." />
                <span asp-validation-for="ReleasePlan" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ReleaseFact" class="control-label"></label>
                <input asp-for="ReleaseFact" class="form-control" />
                <span asp-validation-for="ReleaseFact" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quarter" class="control-label"></label>
                <input asp-for="Quarter" class="form-control" readonly />
                <span asp-validation-for="Quarter" class="text-danger"></span>
```

```
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" readonly />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CompanyId" class="control-label"></label>
                <select asp-for="CompanyId" class="form-control" asp-items="ViewBag.Compa-
nyId"></select>
                <span asp-validation-for="CompanyId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ProductId" class="control-label"></label>
                <select asp-for="ProductId" class="form-control" asp-items="View-
Bag.ProductId"></select>
                <span asp-validation-for="ProductId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Редактировать" class="btn btn-primary" />
            </div>
        </form>
        <form method="post" asp-action="Delete">
            <input type="hidden" asp-for="ReleaseId" />
            <input type="submit" value="Удалить" class="btn btn-danger" />
        </form>
    </div>
</div>

<br/>
<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
    <script type="text/javascript">
    $.validator.methods.range = function (value, element, param) {
        var globalizedValue = value.replace(",", ".");
        return this.optional(element) || (globalizedValue >= param[0] && globalizedValue <=
param[1]);
    }
    $.validator.methods.number = function (value, element) {
        return this.optional(element) || /^-
?(?:\d+|\d{1,3}(?:[\s\.,]\d{3})+)(?:[\.,]\d+)?$/.test(value);
    }
    </script>
}
```

Index.cshtml:

```
@model IndexViewModel<Release>

@addTagHelper "*, CourseWorkDb"

@{
    ViewData["Title"] = "Реализация продукции";
}

<br />
<form asp-action="Index" method="get">
    <div class="form-row col-md-8">
        <div class="col-2">
            <label class="control-label">Предприятие </label>
```

```
        </div>
        <div class="col-4">
            @Html.DropDownList("company", Model.FilterViewModel.Companies as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col-1">
            <label class="control-label">Год </label>
        </div>
        <div class="col-3">
            @Html.DropDownList("year", Model.FilterViewModel.Years as SelectList,
                    htmlAttributes: new { @class = "form-control" })
        </div>
        <div class="col">
            <input type="submit" value="Найти" class="btn btn-secondary" />
        </div>
    </div>
</form>

<br />
<div class="table-responsive">
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col">Предприятие</th>
                <th scope="col">Фактическая реализация продукции</th>
                <th scope="col">План по реализации</th>
                <th scope="col">Год</th>
                <th scope="col">Квартал</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model.Items)
            {
                <tr onclick="window.location.href='/Releases/Details/@item.ReleaseId'; re-
turn false">
                    <td>
                        @Html.DisplayFor(modelItem => item.Company.CompanyName)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.ReleaseFact)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.ReleasePlan)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Year)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Quarter)
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

<page-link page-model="Model.PageViewModel" page-action="Violators"
           page-url-organization="@(Model.FilterViewModel.SelectedCompany)"
           page-url-year="@(Model.FilterViewModel.SelectedYear)"></page-link>
```

Edit.cshtml:

```
@model CourseWorkDb.ViewModels.ChangeRoleViewModel
```

```
@{
    ViewData["Title"] = "Редактирование ролей";
}

<h5>Редактирование ролей пользователя @Model.UserEmail</h5>

<form method="post" asp-controller="Roles" asp-action="Edit" >
    <input type="hidden" name="userId" value="@Model.UserId" />
    <div class="form-group">
        <input type="checkbox" name="adminRole"
               @(Model.UserAdminRole.Equals(Model.AdminRole) ? "checked=\"checked\"" : "")
/>
        @Model.AdminRole
        <br/>
    </div>
    <button type="submit" class="btn btn-primary">Сохранить</button>
</form>
```

_Layout.cshtml:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link href="~/favicon.ico" rel="icon" type="image/x-icon" />
    <title>@ViewData["Title"]</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/lib/user-form.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" asp-controller="Home" asp-action="Index">CourseWork</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-tar-
get="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    @if (User.Identity.IsAuthenticated)
                    {
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown-
MenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Предприятия</a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdownMenu-
Link">
                                <a class="dropdown-item" asp-controller="Companies" asp-ac-
tion="Index">Список</a>
                                <a class="dropdown-item" asp-controller="Companies" asp-ac-
tion="Create">Добавить</a>
                            </div>
                        </li>

                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown-
MenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Виды продук-
ции</a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdownMenu-
Link">
```

```
                                <a class="dropdown-item" asp-controller="ProductTypes" asp-
action="Index">Список</a>
                                <a class="dropdown-item" asp-controller="ProductTypes" asp-
action="Create">Добавить</a>
                            </div>
                        </li>

                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown-
MenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Продукция</a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdownMenu-
Link">
                                <a class="dropdown-item" asp-controller="Products" asp-ac-
tion="Index">Список</a>
                                <a class="dropdown-item" asp-controller="Products" asp-ac-
tion="Create">Добавить</a>
                            </div>
                        </li>

                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown-
MenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Другое</a>
                            <div class="dropdown-menu" aria-labelledby="navbarDropdownMenu-
Link">
                                <h6 class="dropdown-header">Выпуск продукции</h6>
                                <a class="dropdown-item" asp-controller="Outputs" asp-ac-
tion="Index">Список</a>
                                <a class="dropdown-item" asp-controller="Outputs" asp-ac-
tion="Create">Добавить</a>
                                <div class="dropdown-divider"></div>
                                <h6 class="dropdown-header">Реализация продукции</h6>
                                <a class="dropdown-item" asp-controller="Releases" asp-ac-
tion="Index">Список</a>
                                <a class="dropdown-item" asp-controller="Releases" asp-ac-
tion="Create">Добавить</a>
                                <div class="dropdown-divider"></div>
                                <a class="dropdown-item" asp-controller="Companies" asp-ac-
tion="Comparison">Сравнение потребления</a>
                                <a class="dropdown-item" asp-controller="Companies" asp-ac-
tion="Violators">Нарушители норм потребления</a>
                            </div>
                        </li>

                        @if (User.IsInRole(Roles.Admin))
                        {
                            <li class="nav-item dropdown">
                                <a class="nav-link dropdown-toggle" href="#" id="navbar-
DropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-ex-
panded="false">Пользователи</a>
                                <div class="dropdown-menu" aria-labelledby="navbarDropdown-
MenuLink">
                                    <a class="dropdown-item" asp-controller="Users" asp-ac-
tion="Index">Список</a>
                                    <a class="dropdown-item" asp-controller="Users" asp-ac-
tion="Create">Добавить</a>
                                </div>
                            </li>
                        }
                    }
                </ul>
                    <form class="form-inline my-2 my-lg-0" method="post" asp-controller="Ac-
count" asp-action="LogOff">
                        <input class="btn btn-outline-danger" type="submit" value="Выйти" />
                    </form>
```

```
                </div>
            </nav>

        </header>

        <main role="main" class="pb-3" style="margin-left: 1%; margin-right: 1%;">
            @RenderBody()
        </main>

        <footer class="border-top footer text-muted">
            <div class="container">
                &copy; CourseWorkDb 2019
            </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>
```

ChangePAssword.cshtml:

```
@model CourseWorkDb.ViewModels.ChangePasswordViewModel
@{
    ViewBag.Title = "Изменение пароля";
}


<div class="container">
    <div class="row justify-content-md-center">
        <div class="col-md-auto">
            <form class="form-horizontal" method="post" asp-action="ChangePassword" asp-con-
troller="Users">
                <span class="heading">Изменение пароля пользователя</span>
                <div asp-validation-summary="ModelOnly"></div>
                <input type="hidden" asp-for="Id" />
                <div class="form-group">
                    <input type="email" class="form-control" asp-for="Email" place-
holder="Email" required readonly>
                </div>
                <div class="form-group help">
                    <input type="password" class="form-control" asp-for="NewPassword" place-
holder="Пароль" required>
                </div>
                <div class="form-group help">
                    <input type="password" class="form-control" asp-for="NewPasswordConfirm"
placeholder="Повторите пароль" required>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-default">Изменить пароль</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

Create.cshtml:

```
@model CourseWorkDb.ViewModels.CreateUserViewModel
@{
    ViewData["Title"] = "Регистрация";
}
```

```
<div class="container">
    <div class="row justify-content-md-center">
        <div class="col-md-auto">
            <form class="form-horizontal" method="post" asp-controller="Users" asp-ac-
tion="Create">
                <span class="heading">Регистрация нового пользователя</span>
                <div asp-validation-summary="All"></div>
                <div class="form-group">
                    <input type="email" class="form-control" asp-for="Email" place-
holder="Email" required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="Surname" place-
holder="Фамилия" required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="Name" placeholder="Имя"
required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="MiddleName" place-
holder="Отчество" required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="PhoneNumber" place-
holder="Номер телефона" required>
                </div>
                <div class="form-group help">
                    <input type="password" class="form-control" asp-for="Password" place-
holder="Пароль" required>
                </div>
                <div class="form-group help">
                    <input type="password" class="form-control" asp-for="PasswordConfirm"
placeholder="Повторите пароль" required>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-default">Зарегистрировать</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

Edit.cshtml:

```
@model CourseWorkDb.ViewModels.EditUserViewModel
@{
    ViewBag.Title = "Редактирование пользователя";
}

<div class="container">
    <div class="row justify-content-md-center">
        <div class="col-md-auto">
            <form class="form-horizontal" method="post" asp-action="Edit" asp-control-
ler="Users">
                <span class="heading">Редактирование пользователя</span>
                <div asp-validation-summary="ModelOnly"></div>
                <input type="hidden" asp-for="Id" />
                <div class="form-group">
                    <input type="email" class="form-control" asp-for="Email" place-
holder="Email" required>
                </div>
                <div class="form-group">
```

```
                    <input type="text" class="form-control" asp-for="Surname" place-
holder="Фамилия" required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="Name" placeholder="Имя"
required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="MiddleName" place-
holder="Отчество" required>
                </div>
                <div class="form-group">
                    <input type="text" class="form-control" asp-for="PhoneNumber" place-
holder="Номер телефона" required>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-default">Изменить</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

Index.cshtml:

```
@model IEnumerable<CourseWorkDb.Models.Authentication.User>
@{
    ViewBag.Title = "Список пользователей";
}

<div class="table-responsive">
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Email</th>
                <th scope="col">Фамилия</th>
                <th scope="col">Имя</th>
                <th scope="col">Отчество</th>
                <th></th>
            </tr>
        </thead>

        @foreach (var user in Model)
        {
            <tr>
                <td>@user.Email</td>
                <td>@user.Surname</td>
                <td>@user.Name</td>
                <td>@user.MiddleName</td>
                <td>
                    <form asp-action="Delete" asp-route-id="@user.Id" method="post">
                        <a class="btn btn-sm btn-primary" asp-action="Edit" asp-route-
id="@user.Id">Редактировать</a>
                        <a class="btn btn-sm btn-primary" asp-action="ChangePassword" asp-
route-id="@user.Id">Сменить пароль</a>
                        @if (!user.UserName.Equals(User.Identity.Name))
                        {
                            <a class="btn btn-sm btn-primary" asp-controller="Roles" asp-
action="Edit" asp-route-userid="@user.Id">Права доступа</a>
                            <button type="submit" class="btn btn-sm btn-dan-
ger">Удалить</button>
                        }
                    </form>
                </td>
```

```
            </tr>
        }
    </table>
</div>
```

Program.cs:

```
using CourseWorkDb.Models.Authentication;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Threading.Tasks;

namespace CourseWorkDb
{/*
    public class Program
    {
        public static async System.Threading.Tasks.Task Main(string[] args)
        {
            IHost host = CreateHostBuilder(args).Build();

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    var userManager = services.GetRequiredService<UserManager<User>>();
                    var rolesManager = services.GetRequiredService<RoleManager<Identi-
tyRole>>();
                    await RoleInitializer.InitializeAsync(userManager, rolesManager);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred while seeding the database.");
                }
            }

            host.Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }*/

    public class Program
    {
        public static async Task Main(string[] args)
        {
            //BuildWebHost(args).Run();
            var host = BuildWebHost(args);

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
```

```
                {
                    var userManager = services.GetRequiredService<UserManager<User>>();
                    var rolesManager = services.GetRequiredService<RoleManager<Identi-
tyRole>>();
                    await RoleInitializer.InitializeAsync(userManager, rolesManager);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred while seeding the database.");
                }
            }

            host.Run();
        }

        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args).UseStartup<Startup>().Build();


    }
}
```

Startup.cs:

```
using CourseWorkDb.Middleware;
using CourseWorkDb.Models;
using CourseWorkDb.Models.Authentication;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;

namespace CourseWorkDb
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            string connection1 = Configuration.GetConnectionString("SQLServerConnection");
            string connection2 = Configuration.GetConnectionString("UsersSQLServerConnec-
tion");

            services.AddDbContext<IdentityContext>(options =>
                options.UseSqlServer(connection2));

            services.AddIdentity<User, IdentityRole>()
                .AddEntityFrameworkStores<IdentityContext>()
                .AddErrorDescriber<RussianIdentityErrorDescriber>();

            services.AddDbContext<RationingDbContext>(options =>
```

```
                options.UseSqlServer(connection1));

            services.AddDistributedMemoryCache();
            services.AddSession(options =>
            {
                options.IdleTimeout = TimeSpan.FromMinutes(60);
                options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
                options.Cookie.SameSite = SameSiteMode.Strict;
                options.Cookie.HttpOnly = true;
                options.Cookie.IsEssential = true;
            });

            services.AddControllersWithViews();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseInitializerMiddleware();
            app.UseStatusCodePagesWithRedirects("/");
            app.UseSession();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Чертёж структуры *web*-приложения