Допущен к защите:

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ года

Дипломный проект защищен:

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_года

Зав. учебной частью \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

##### ДИПЛОМНИЙ ПРОЕКТ

**КА «Шаг» ППС 24-15. ПСТ. 06/2015. 11030000906**

Тема:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(на русском языке)

Тема:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(на английском языке)

Слушатель \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О на русском языке)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О на английском языке)

Руководитель \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(рекомендуемая оценка)

Экзаменационная комиссия по защите дипломных проектов:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата) (Ф.И.О)

Мариуполь 2018

ЗАДАНИЕ И КАЛЕНДАРНЫЙ ПЛАН

по выполнению дипломной работы

|  |  |  |
| --- | --- | --- |
| студентом группы |  |  |
|  | | |
| *(Фамилия, Имя, Отчество)* | | |

|  |  |
| --- | --- |
| Тема работы: |  |
|  | |
|  | |

Утверждена приказом директора филиала от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ № \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок сдачи студентом законченной работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем дипломной работы: 50-80 машинописных страниц.

Перечень прилагаемого материала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Организации, на базе которых выполняется дипломная работа: |
| *а)* |
| *б)* |
| *в)* |

##### *Календарный план выполнения дипломной работы*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование этапа работ | Срок выполнения | | Оценка выполненного этапа работы | Роспись руководителя |
| по плану | фактически |
| Написание введения |  |  |  |  |
| Написание главы 1 |  |  |  |  |
| Написание главы 2 |  |  |  |  |
| Написание главы 3 |  |  |  |  |
| Написание заключения |  |  |  |  |
| Оформление дипломной работы |  |  |  |  |
| Получение отзыва руководителя |  |  |  |  |
| Подготовка доклада и наглядных пособий |  |  |  |  |
| Защита дипломной работы |  |  |  |  |

Оглавление

[Введение 5](#_Toc511732042)

[1 Подготовка к проектированию 7](#_Toc511732043)

[1.1 Описание требований к проекту 7](#_Toc511732044)

[1.2 Выбор технологий для реализации проекта 7](#_Toc511732045)

[2 Проектирование и реализация приложения 14](#_Toc511732046)

[2.1 Разработка базы данных 14](#_Toc511732047)

[2.2 Разработка веб-приложения 17](#_Toc511732048)

[2.3 Система авторизации пользователей 19](#_Toc511732049)

[3 Руководство пользователя 21](#_Toc511732050)

[3.1 Работа с приложением 21](#_Toc511732051)

[Заключение 26](#_Toc511732052)

[Список использованной литературы 28](#_Toc511732053)

[Приложения 29](#_Toc511732054)

[Приложение №1 29](#_Toc511732055)

[Приложение №2 30](#_Toc511732056)

[Приложение №3 36](#_Toc511732057)

[Приложение №4 37](#_Toc511732058)

[Приложение №5 47](#_Toc511732059)

[Приложение №6 59](#_Toc511732060)

[Приложение №7 73](#_Toc511732061)

[Лист замечаний 90](#_Toc511732062)

# Введение

Дипломная работа на тему: CTFinder (Creative Task Finder) - веб-приложение для поиска развивающих заданий. Объем работы 50 – 80 страниц.

Цель работы: разработать веб-приложение согласно теме .

Огромное число людей сегодня хотят работать так, чтобы труд не только обеспечивал их материально, но также приносил им удовольствие и создавал настоящую полезность для других. Многие преследуют эту же цель в не занятое основной работой время: создают и выкладывают в свободный доступ в Интернет полезный контент, участвуют в проектах по разработке открытого программного обеспечения, помогают больным и нуждающимся, осваивают новые сферы знаний и навыков, чтобы улучшить жизнь. Многие студенты ищут такие задания для дипломных работ, чтобы их результаты принесли практическую пользу и могли служить портфолио при устройстве на работу.

Для всех этих случаев предназначается CTFinder — веб-приложение, при помощи которого авторы заданий и желающие их выполнять с целью, прежде всего, саморазвития и создания полезности для других, как можно скорее находили друг друга.

Приложение должно быть доступным везде, где есть выход в Интернет, и поддерживать полную локализацию ресурсов на любые языки мира. Оно должно быть изначально рассчитано на высокие нагрузки, а хостинг должен обеспечивать легкую масштабируемость.

Всемирно известных конкурентов с описанной выше культурной спецификой в Интернете пока нет, но если они появятся — это только к лучшему, так как важен сам прогрессивный принцип нашего приложения.

Приложение размещается на хостинге в бесплатном режиме. Когда потребуется больше ресурсов, будет добавлена контекстная реклама и опубликовано предложение поддержать проект материально.

# 1 Глава

## Описание требований к проекту

Написать веб-приложение для автоматизации учета, заказа и списания запчастей на судах торгового флота. Для работы с программой достаточно базовых навыков работы с компьютером, обслуживание и наладку приложения должен производить системный администратор. Информация о запчастях и другая служебная информация хранится в единой базе данных. Разграничены права доступа к функциям приложения в зависимости от занимаемой должности. Отсутствует возможность неавторизированного использования. Возможность создания ордера на заказ запчастей. Учет плановых и внеплановых работ на оборудовании. Просмотр перечня плановых технических обслуживаний.

## 1.2 Выбор технологий для реализации проекта

Реализовываться проект будет на технологии ASP.NET MVC, и для наилучшей совместимости база данных будет Microsoft SQL Server.

ASP.NET (Active Server Pages для .NET) — технология создания веб-приложений и веб-сервисов от компании Майкрософт. Она является составной частью платформы Microsoft .NET и развитием более старой технологии Microsoft ASP.

Хотя ASP.NET берёт своё название от старой технологии Microsoft ASP, она значительно от неё отличается. Microsoft полностью перестроила ASP.NET, основываясь на Common Language Runtime (CLR), которая является основой всех приложений Microsoft .NET. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET и JScript .NET). ASP.NET имеет преимущество в скорости по сравнению со скриптовыми технологиями, так как при первом обращении код компилируется и помещается в специальный кэш, и впоследствии только исполняется, не требуя затрат времени на парсинг, оптимизацию, и т. д.

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

Свою реализацию паттерна представляет платформа ASP.NET MVC. 2013 год ознаменовался выходом новой версии ASP.NET MVC - MVC 5, а также релизом Visual Studio 2013, которая предоставляет инструментарий для работы с MVC5.

Хотя во многих аспектах MVC 5 не слишком сильно будет отличаться от MVC 4, многое из одной версии вполне применимо к другой, но в то же время есть и существенные отличия:

В MVC 5 изменилась концепция аутентификации и авторизации. Вместо SimpleMembershipProvider была внедрена система ASP.NET Identity, которая использует компоненты OWIN и Katana.

Для большинства разработчиков ASP.NET платформа Identity является первым знакомством с архитектурным шаблоном Open Web Interface for .NET (OWIN). OWIN – это уровень абстракции, который изолирует веб-приложения из среды, в которой они размещены. Идея заключается в том, что такая абстракция позволит добиться больших возможностей в стеке технологий ASP.NET, большей гибкости в среде разработки веб-приложений и облегченной разработки серверной инфраструктуры приложений.

OWIN - это открытый стандарт (с которым более подробно вы можете ознакомиться по ссылке owin.org). Microsoft создала проект Katana Project, представляющий наглядную реализацию стандарта OWIN и включающий набор компонентов, которые обеспечивают функциональность веб-приложений. В приложении OWIN/Katana Microsoft наглядно продемонстрировали изоляцию стека технологий ASP.NET от остальной платформы .NET Framework.

Используя OWIN, разработчики могут подключать только те компоненты, которые нужны прямо здесь и сейчас, а не работать с целой платформой, как это сейчас происходит с ASP.NET. Благодаря такому подходу разрабатываемое приложение не будет перегружено избыточным функционалом, а, следовательно, будет работать быстрее (в теории).

Наглядным примером компонентов, разработанных по принципу OWIN являются библиотеки Web API и SignalR, которые не требуют наличия пространства имен System.Web или работающего сервера IIS для обработки HTTP-запросов. Платформа ASP.NET MVC Framework, в отличие от этих библиотек, зависит от стандартной платформы ASP.NET.

OWIN и Katana пока не имеют серьезного влияния на MVC Framework, но появление такой платформы, как ASP.NET Identity, являющейся полноценным компонентом OWIN, говорит о том, что в скором времени такое поведение может измениться.

Для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap

Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Основные инструменты Bootstrap:

* Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140 px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.
* Шаблоны — фиксированный или резиновый шаблон документа.
* Типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.
* Медиа — представляет некоторое управление изображениями и видео.
* Таблицы — средства оформления таблиц, вплоть до добавления функциональности сортировки.
* Формы — классы для оформления форм и некоторых событий, происходящих с ними.
* Навигация — классы оформления для табов, вкладок, страничности, меню и панели инструментов.
* Алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Свидетельством того, что Bootstrap это очень удачная разработка, является его непрерывно растущая популярность. Нынешний уровень развития этого фреймворка уже позволяет полностью создать любой web-интерфейс.

Добавлены фильтры аутентификации, а также появилась функциональность переопределения фильтров

В MVC 5 также добавлены атрибуты маршрутизации

Это наиболее важные нововведения в MVC 5. Кроме того, есть еще ряд менее значимых, например, использование по умолчанию Entity Framework 6, некоторые изменения при создании проекта (концепция One ASP.NET), дополнительные компоненты и т.д.

В любом случае все полученные при работе с MVC 4 навыки можно успешно применять при использовании MVC 5, учитывая, конечно, нововведения.

“Microsoft SQL Server 2016” это система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Для данного проекта можно использовать бесплатный вариант “Microsoft SQL Server 2016 Express”, его возможностей достаточно, а при необходимости можно легко обновиться на более продвинутую платную версию.

Для удобства работы с базой данных в проекте используется Entity Framework. Entity Framework является продолжением технологии Microsoft ActiveX Data и предоставляет возможность работы с базами данных через объектно-ориентированный код C#. Этот подход предоставляет ряд существенных преимуществ: вам не нужно беспокоиться о коде доступа к данным, вам не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого вы работаете с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов, который в ADO.NET раньше нужно было вставлять в код C# в виде команд, заменен на более удобный подход с LINQ. Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

При работе с Entity Framework вам предоставляются огромные возможности по созданию модели базы данных с помощью интегрированной среды разработки (IDE) Visual Studio. Начиная с версии Entity Framework 4.1 вам предоставляется три подхода по проектированию базы данных, из которых вы можете выбрать для себя подходящий:

Database-First - подходит для проектировщиков баз данных - сначала вы создаете базу данных с помощью различных инструментов (например, SQL Server Management Studio), а затем генерируете EDMX-модель базы данных (предоставляет удобный графический интерфейс для взаимодействия с базой данных в виде диаграмм и объектную модель в виде классов C#). В данном случае вам нужно работать с SQL Server и хорошо знать синтаксис T-SQL, но при этом не нужно разбираться в C#.

Model-First - подходит для архитекторов - сначала вы создаете графическую модель EDMX в Visual Studio (в фоновом режиме создаются классы C# модели), а затем генерируете на основе диаграммы EDMX базу данных. При данном подходе не нужно знать ни деталей T-SQL ни синтаксиса C#.

Code-First - подходит для программистов - при данном подходе модель EDMX вообще не используется, и вы вручную настраиваете классы C# объектной модели (данный подход поддерживает как генерацию сущностных классов из существующей базы данных, так и создание базы данных из созданной вручную модели объектов C#). Очевидно, что это подходит для программистов, хорошо знакомых с синтаксисом C#.

# 2 Глава

## 2.1 Разработка базы данных

База данных приложения состоит из следующих таблиц:

* Departments – в этой таблице хранится информация о организационных подразделениях судна.
* Maintenance – здесь хранится информация о всех проделанных работах, персонале их проводившем, использованных запасных частях, дате и количестве часов работы оборудования на момент ремонта.
* Manufacturers – эта таблица содержит информацию о производителях оборудования, наименование компании и ее контактную информацию.
* Measurements – здесь хранятся единицы измерения.
* Orders – в этой таблице хранятся заявки на поставку запасных частей, их перечень и даты создания заявки и ее выполнения.
* PlannedMaintenances – эта таблица заполняется исключительно программным способом при помощи триггеров в таблицах Plants и Maintenance. Она содержит информацию о сроках планового технического обслуживания оборудования.
* Plants – здесь содержится информация об имеющимся оборудовании, его дате производства, типе, серийном номере, количестве наработанных часов, производителе, департаменте который им заведует.
* PlantsMaintenances – эта таблица содержит перечень плановых (регулярных) технических обслуживаний оборудования, их интервал и принадлежность к оборудованию.
* Spareparts – в этой таблице хранится перечень запасных частей, ссылки на чертеж, номер каталога, позиция на чертеже, количество на складе и в механизме, единица измерения, механизм к которому она относится.

Для хранения перечней использованных, а также заказанных запасных частей я буду использовать тип данных XML, так я сокращу количество таблиц в базе данных и облегчу экспорт/импорт этих данных в другие приложения. Схема базы данных приведена в Приложение №1.

Чтобы заполнять и обновлять данные в таблице PlannedMaintenances необходимо написать три триггера первый будет в таблице PlantsMaintenances, он будет добавлять или обновлять запись в PlannedMaintenances при добавлении новой записи или обновлении существующей в таблице PlantsMaintenances. Вот его код:

CREATE TRIGGER [dbo].[UpdateMaintenances\_plantsM]

ON [dbo].[PlantsMaintenances]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @x int

IF (EXISTS (

SELECT [dbo].[Maintenance].[current\_rh\_when\_done]

FROM [dbo].[Maintenance], [inserted]

WHERE [dbo].[Maintenance].[maintenance\_id] = [inserted].[maintenanceid]))

SET @x = (

SELECT [dbo].[Maintenance].[current\_rh\_when\_done]

FROM [dbo].[Maintenance], [inserted]

WHERE [dbo].[Maintenance].[maintenance\_id] = [inserted].[maintenanceid])

ELSE

SET @x = (

SELECT [pl].[running\_hours]

FROM [dbo].[Plants] AS [pl], [inserted] AS [i]

WHERE [i].[plantid] = [pl].[plantid])

IF ((SELECT COUNT(\*) FROM [deleted])= 0)

INSERT [dbo].[PlannedMaintenances]([maintenanceid], [last\_service\_rh], [planned\_service\_rh], [servise\_overlap], [plantid])

SELECT [i].[maintenanceid], @x, (@x + [i].[sevice\_intervals]),

(@x + [i].[sevice\_intervals]) - [p].[running\_hours], [i].[plantid]

FROM [dbo].[PlantsMaintenances] AS [pl], [inserted] AS [i], [Plants] AS [p]

WHERE [pl].[maintenanceid] = [i].[maintenanceid] AND [i].[plantid] = [p].[plantid]

ELSE

UPDATE [dbo].[PlannedMaintenances]

SET [last\_service\_rh] = (CASE WHEN [m].[current\_rh\_when\_done] <> NULL THEN [m].[current\_rh\_when\_done]

WHEN [m].[current\_rh\_when\_done] = NULL THEN [p].[running\_hours] END),

[planned\_service\_rh] = [last\_service\_rh] + [i].[sevice\_intervals],

[servise\_overlap] = [planned\_service\_rh] - [p].[running\_hours],

[plantid] = [i].[plantid]

FROM [inserted] AS [i], [dbo].[PlantsMaintenances] AS [pm], [dbo].[Plants] AS [p], [dbo].[Maintenance] AS [m]

WHERE [pm].[maintenanceid] = [i].[maintenanceid] AND [p].[plantid] = [i].[plantid] AND [i].[maintenanceid] = [m].[maintenance\_id]

END

Теперь необходимо написать триггеры, которые бы обновляли сроки планового ремонта при обновлении сведений о текущих часах наработки оборудования или проведенном техобслуживании. Они будут располагаться в таблицах Maintenance и Plants вот их программный код:

CREATE TRIGGER [dbo].[PlannedMaintenances\_Update]

ON [dbo].[Maintenance]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF((SELECT COUNT(\*) FROM [deleted]) = 0)

UPDATE [pm]

SET [last\_service\_rh] = [i].[current\_rh\_when\_done],

[planned\_service\_rh] = [i].[current\_rh\_when\_done] + [pms].[sevice\_intervals],

[servise\_overlap] = [planned\_service\_rh] - [p].[running\_hours]

FROM [dbo].[PlannedMaintenances] AS [pm], [inserted] AS [i], [dbo].[Plants] AS [p], [dbo].[PlantsMaintenances] AS [pms]

WHERE [pm].[maintenanceid] = [i].[maintenance\_id] AND [i].[plantid] = [p].[plantid] AND [pms].[maintenanceid] = [i].[maintenance\_id]

ELSE

IF(EXISTS(SELECT \* FROM [deleted] AS [d], [inserted] AS [i] WHERE [i].[maintenanceid] = [d].[maintenanceid] AND [i].[current\_rh\_when\_done] <> [d].[current\_rh\_when\_done]))

UPDATE [pm]

SET [last\_service\_rh] = [i].[current\_rh\_when\_done],

[planned\_service\_rh] = [i].[current\_rh\_when\_done] + [pms].[sevice\_intervals],

[servise\_overlap] = [planned\_service\_rh] - [p].[running\_hours]

FROM [dbo].[PlannedMaintenances] AS [pm], [inserted] AS [i], [dbo].[Plants] AS [p], [dbo].[PlantsMaintenances] AS [pms]

WHERE [pm].[maintenanceid] = [i].[maintenance\_id] AND [i].[plantid] = [p].[plantid] AND [pms].[maintenanceid] = [i].[maintenance\_id]

ELSE

PRINT ''Nothing to update''

END

CREATE TRIGGER [dbo].[Update\_service\_intervals]

ON [dbo].[Plants]

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

UPDATE pm

SET [servise\_overlap] = [pm].[planned\_service\_rh] - [i].[running\_hours]

FROM [dbo].[PlannedMaintenances] AS pm, [inserted] AS i

WHERE [pm].[plantid] = [i].[plantid]

END

Отношения между таблицами, типы данных наименования строк вся эта информация указана на схеме базы данных и присутствует в скрипте ее создания на прилагающимся носителе.

## 2.2 Разработка веб-приложения

Веб-приложение будет разрабатываться на основе ASP.NET MVC 5. Для работы с базой данных будет использоваться Entity Framework 6, оформление представлений будет реализовано с помощью Bootstrap 4, скрипты будут написаны с применением JQuery. Для создания проекта использую Microsoft Visual Studio 2017.

После создания в Visual Studio ASP.NET MVC 5 проекта с авторизацией типа Individual User Accounts мастер создаст стандартную структуру с уже имеющимся некоторым функционалом. Теперь необходимо сгенерировать сущности из уже разработанной базы данных и добавить их в проект. Для этого используем мастер, ADO.NET EDM генерирующий классы сущностей из базы данных. В результате мы получим класс с расширением .edmx содержащий диаграмму базу данных со связями между таблицами, классы сущностей (таблиц) класс DbContext и другие классы.

Для удобства работы с валидацией в формах, добавлении дополнительных методов к классам сущностей нашей базы данных необходимо создать два класса, первый будет содержать классы сущностей с необходимыми аннотациями назовем его Metadata.cs (программный код его приведен в Приложение №2), а второй назовем PartialClasses.cs (программный код его приведен в Приложение №3) в нем мы при помощи аннотаций вида: [MetadataType(typeof(MaintenanceMetadata))] ассоциируем наши классы с аннотациями из Metadata.cs с классами соответствующих сущностей базы данных. Теперь при необходимости можно заново генерировать мастером классы сущностей без потери всех написанных нами дополнительных методов и указанных аннотаций.

Теперь необходимо создать контроллеры к каждой сущности из базы данных. Мастером создаем контроллеры. В процессе генерации мастер создает и представления для каждого метода контроллера. Методы стандартные позволяющие просматривать, добавлять, редактировать и удалять строки из одноименной таблицы базы данных. Поскольку мастер генерирует стандартный код его функциональности недостаточно. Например, таблица Spareparts может содержать тысячи строк вывод которых может занять много времени и поиск необходимой строки весьма затруднителен в таком «лесу» поэтому необходимо реализовать функционал поиска, сортировки и постраничного вывода.

Постраничный вывод организуем при помощи утилиты X.PagedList добавив ее в проект при помощи NuGet менеджера. Для ввода текста для поиска добавим необходимые элементы в представление, также необходимо будет добавить метод в контроллер для показа чертежа и скрипты для управления элементами поиска и сортировки (полный листинг кода контроллера и его представлений приведен в Приложение №4).

Таблица Maintenance содержит XML поле, для хранения списка использованных запчастей. Создадим контроллер UsedSparepartController для работы с этим списком и представления с моделью данных. Работа с XML в контроллере будет реализована через стандартный класс XmlDocument и его методы. Программный код модели контроллера и представлений показан в Приложение №5.

Таблица Orders тоже содержит XML поле, для хранения списка заказанных запасных частей поэтому создадим контроллер OrderedSparepartController и соответствующую модель OrderedSparepartModel ну и соответственно представления для отображения пользователю. Программный код модели контроллера и представлений показан в Приложение №6. Добавим дополнительные методы в контроллер OrdersController в которых реализуем трансформацию XML данных в PDF используя для этого XSL шаблон и NuGet пакет Select.HtmlToPdf. Для конвертации придётся сначала конвертировать XML в HTML используя XSL шаблон и затем конвертировать готовый HTML в PDF при помощи Select.HtmlToPdf. Программный код контроллера и представлений в Приложение №6.

Контроллеры таблиц Departments, Manufacturers, Plants, PlantsMaintenances, Measurements состоят из созданного мастером кода за исключением добавленных аннотаций с правами доступа к различным методам. Представления этих контроллеров также имеют стандартно сгенерированный код.

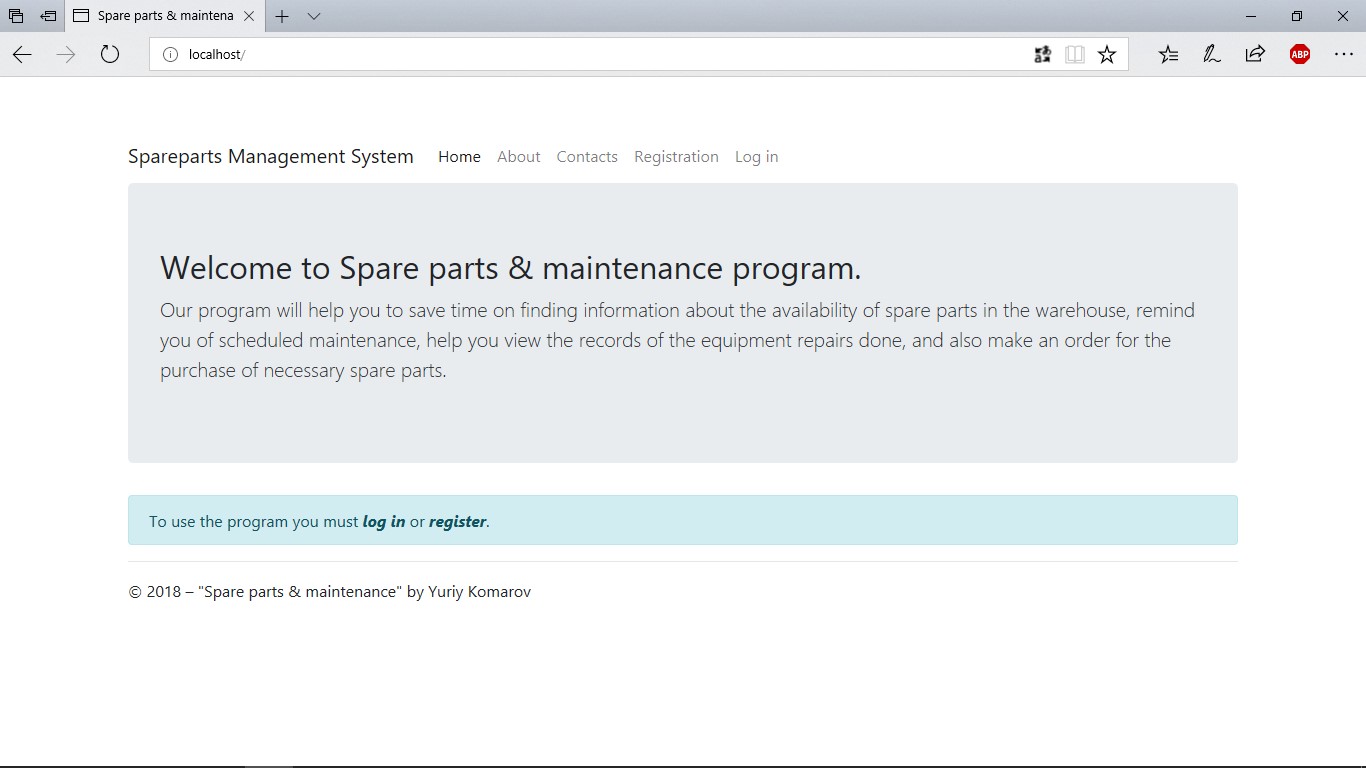
## 2.3 Система авторизации пользователей

Авторизация пользователей реализована средствами ASP.NET Identity. В ходе создания проекта мы указывали в мастере что будем применять идентификацию пользователей на основе индивидуальных аккаунтов, и система сгенерировала уже готовый базовый функционал полностью готовый к работе. Большинство необходимых методов находятся в контроллерах AccountController и ManageController. Так как стандартные поля класса ApplicationUser не достаточны то мы добавим необходимые нам свойства в этот класс. Также придется отредактировать соответствующие методы в контроллерах и представления. Данные будут храниться на том же сервере где и основная база данных, поэтому необходимо изменить стандартную строку подключения ASP.NET Identity в файле web.config с именем DefaultConnection и внести в нее необходимые данные. Листинг программного кода приведен в Приложение №7.

# 3 Глава

## 3.1 Работа с приложением

Работа с приложением построена на основе диалога с пользователем. Начиная с главной страницы пользователю предлагается авторизоваться или пройти регистрацию об этом напоминает информационный баннер, он виден если пользовать не авторизовался рис. 3.1.



После авторизации в строке навигационного меню появляется приветствие, щелчок по которому перенаправляет пользователя на страницу смены пароля его аккаунта. Так же становится доступным пункт меню “Working Tasks” для работы с основным функционалом приложения рис 3.2.

Рисунок 3.1

В выпадающем списке меню “Working Tasks” пользователь видит названия таблиц, по которым легко догадаться что пользователь увидит, перейдя по любой из этих ссылок.

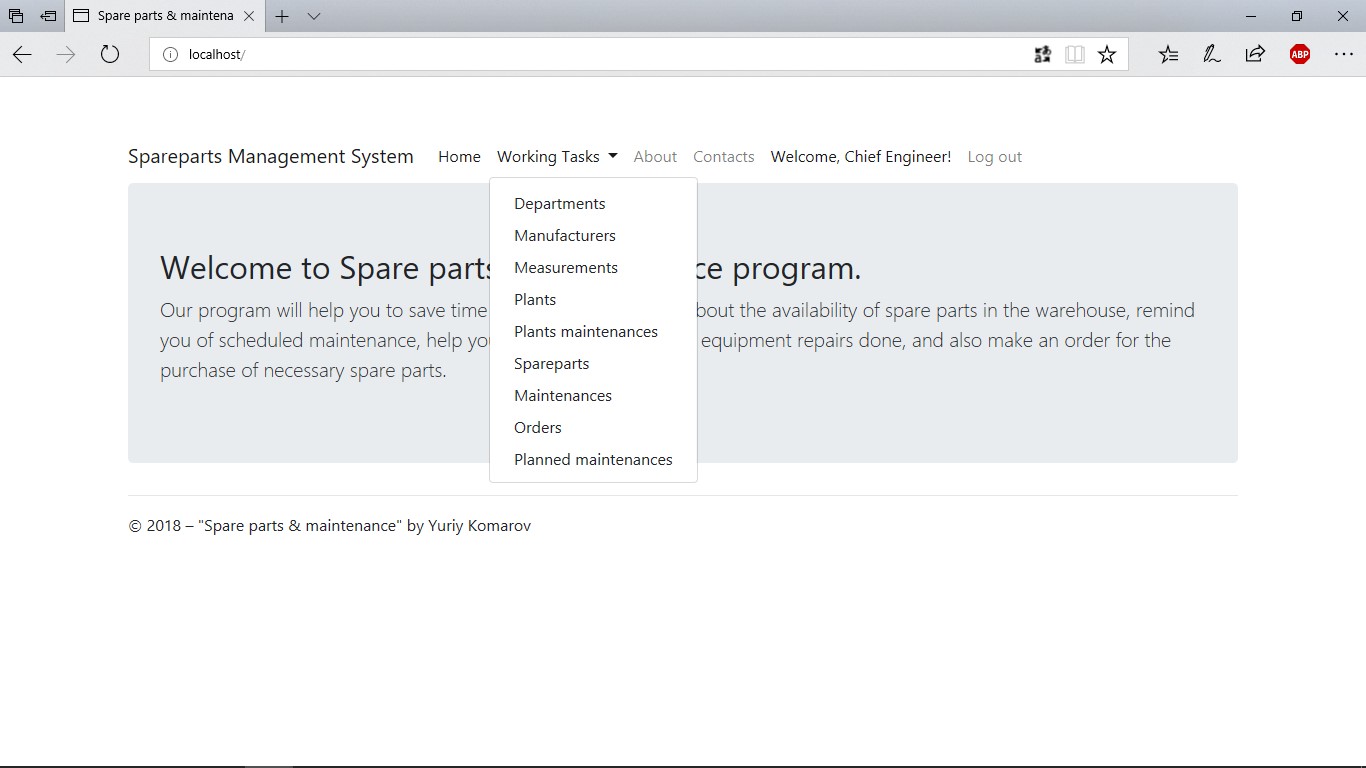


Рисунок 3.2

Работа с таблицами начинается с отображения ее содержания где в крайне правой колонке находятся ссылки на редактирование просмотр или удаление каждой строки отдельно. Над списком присутствует ссылка на создание новой записи в таблице рис 3.3.

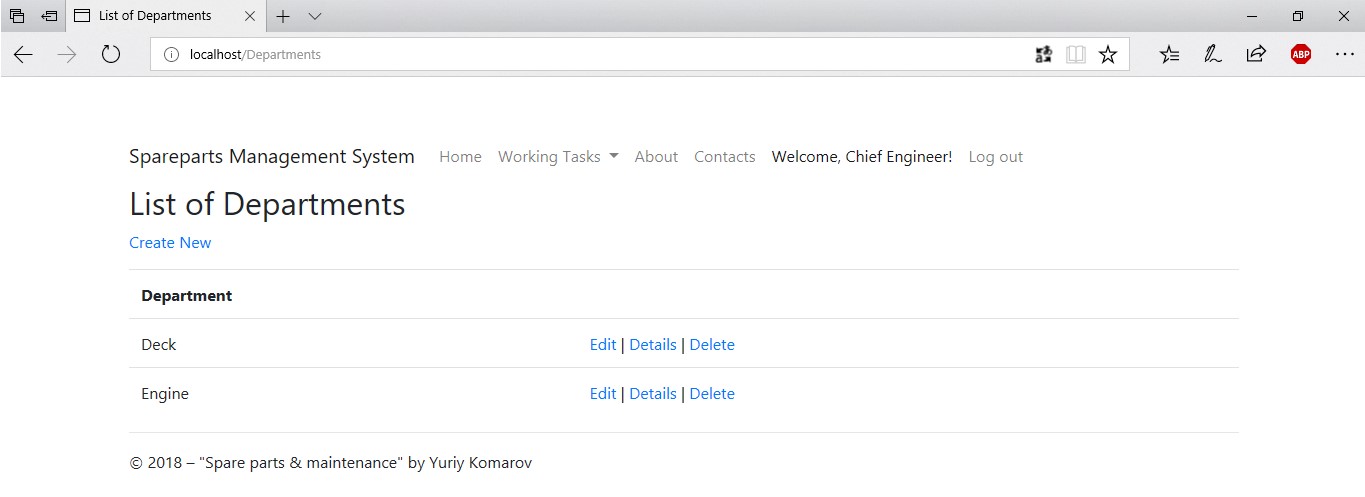


Рисунок 3.3

Из особенностей выделяются три таблицы это Orders Maintenances и Spareparts. Таблицы Orders и Maintenances содержат списки запасных частей, которые хранятся в виде XML строк. Работа с этими списками осуществляется переходом по ссылке “View” если список существует, либо “Create” для его создания.

При переходе по ссылке Orders мы увидим список ордеров из одноименной таблицы, но в каждой строке будет присутствовать дополнительное поле “Get as PDF”. Переход по этой ссылке вызовет скачивание сгенерированного файла заявки на поставку запчастей соответствующего ордера. Дальнейшая работа с этим файлом осуществляется пользователем при помощи любого программного обеспечения, поддерживающего работу с форматом PDF рис 3.4.

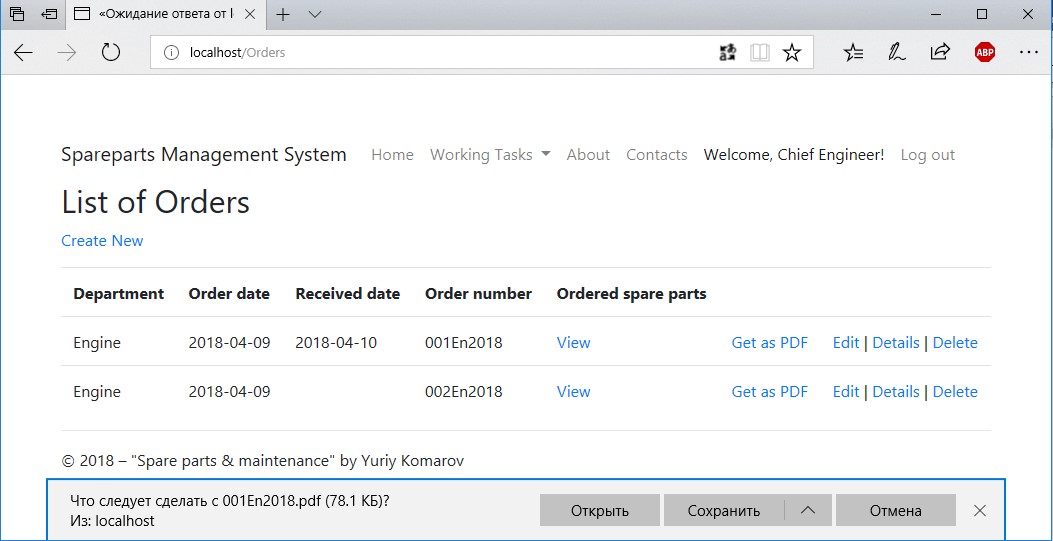


Рисунок 3.4

Таблица Spareparts самая большая в базе данных в ней могут находиться тысячи строк каждая из которых соответствует конкретной запчасти. Для удобства поиска необходимой запчасти в этом списке реализована форма поиска, расположенная над списком рис 3.5. Изначально поле поиска “Search” не активно, для его активации необходимо в выпадающем списке “Search field” выбрать необходимый столбец значения которого будут сравниваться с содержанием поля “Search”. Сброс в начальное состояние осуществляется выбором позиции “None”.

Еще в этой таблице реализована сортировка данных по убыванию/возрастанию по тем же столбцам, как и поиск, заголовки этих столбцов отмечены голубым цветом. Активация сортировки и изменение ее направления осуществляется нажатием на заголовок соответствующего столбца таблицы.

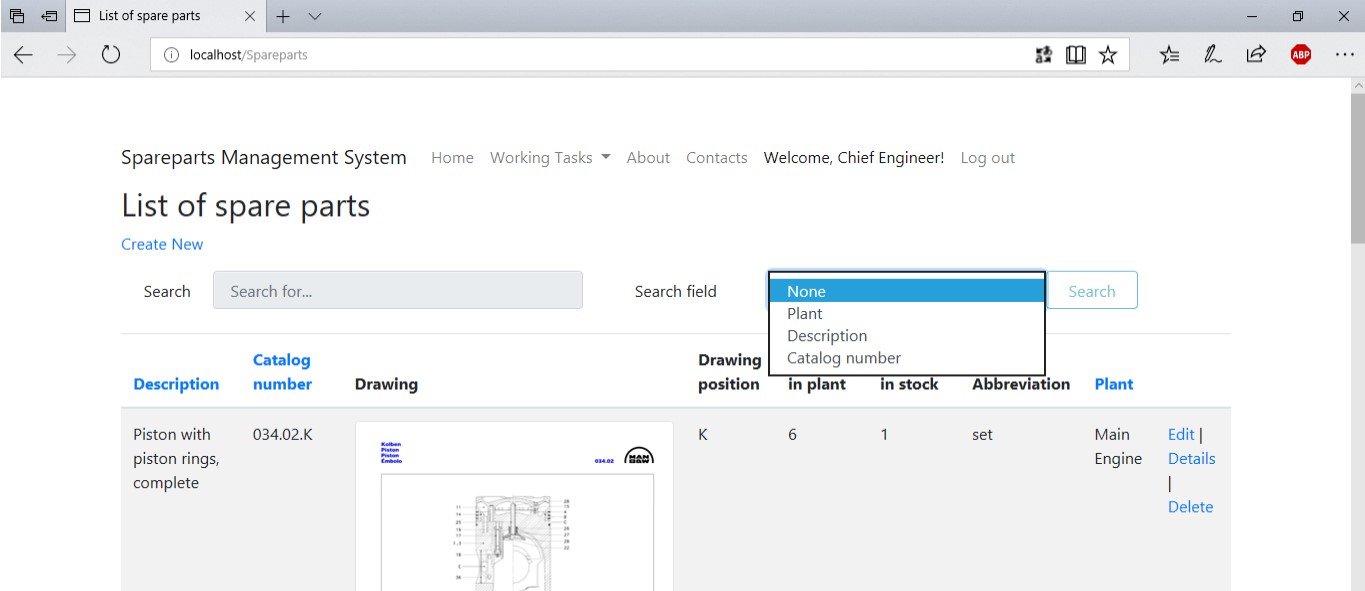


Рисунок 3.5

Нажатие на картинку чертежа откроет страницу с отображением его в 100% величины страницы. Также при создании или редактировании строки таблицы доступна загрузка нового изображения чертежа или его замена.

Таблица “Planned Maintenances” отображает информацию о планируемых технических обслуживаниях оборудования рис. 3.6. Список оборудования формируется в зависимости от департамента в котором работает пользователь. Строки в таблице подсвечиваются разными цветами в зависимости от оставшегося времени до планового техобслуживания, если срок более 48 часов подсвечивается зеленым, от 0 до 48 часов оранжевым, просрочено – красным.

Данная таблица заполняется автоматически на основе данных из трех таблиц “PlantsMaintenances”, “Maintenances” и “Plants”. Редактирование или добавление данных пользователями не предусмотрено.

Все формы для ввода или редактирования информации снабжены проверкой вводимой информации. В случае ввода текста, не соответствующего критериям проверки, выводится соответствующее предупреждение. Форма не будет отправлена на обработку пока будет содержать ошибки рис 3.7.

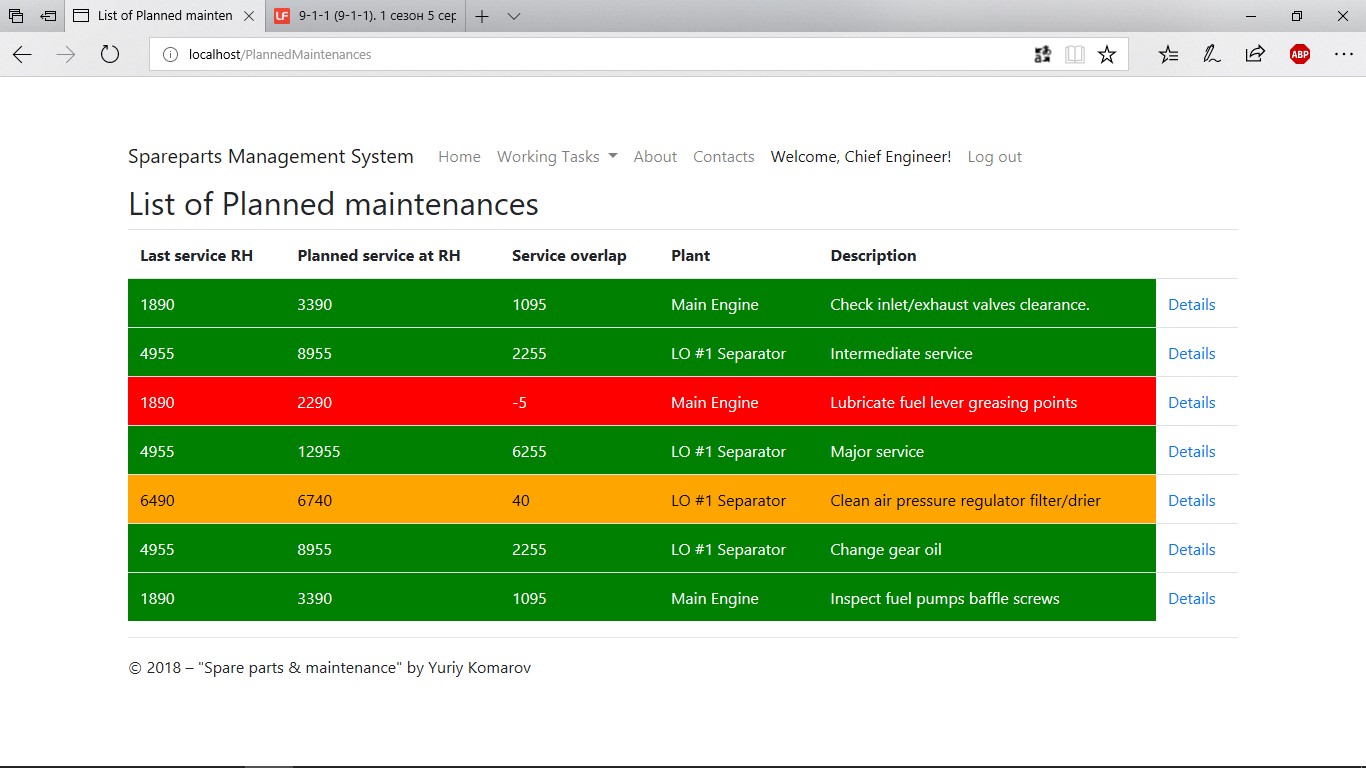


Рисунок 3.6

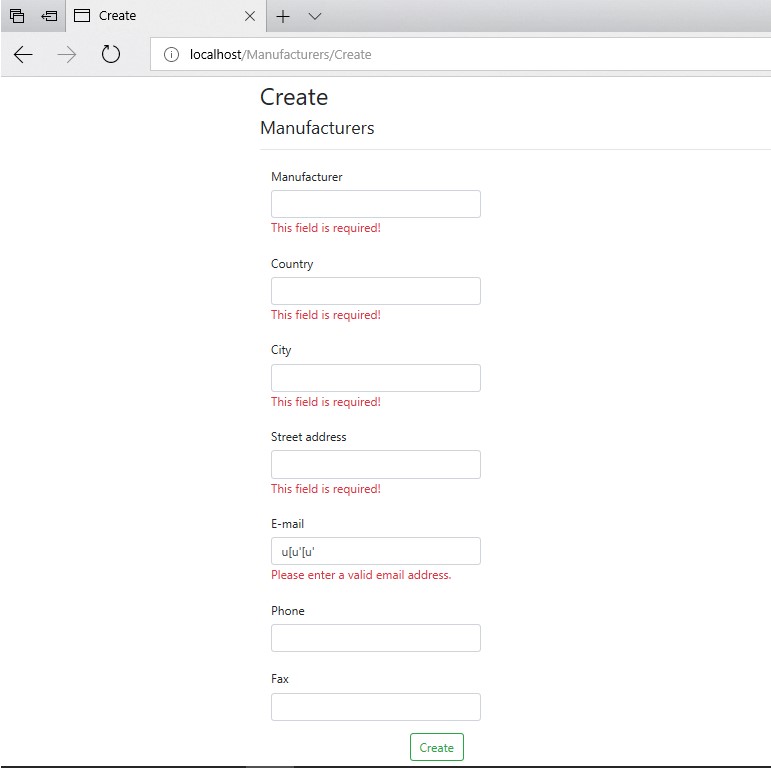


Рисунок 3.7

# Заключение

В результате проделанной работы у меня получилось разработать полнофункциональное веб приложение согласно поставленным задачам.

База данных приложения состоит из таблиц, именованных соответственно реальным элементам оборудования и планирования, поэтому интуитивно понятно, что содержится в каждой конкретной таблице. Связи между таблицами при помощи внешних и внутренних ключей обеспечивают целостность данных при их добавлении или редактировании, а также предотвращают случайное удаление зависимых данных.

Веб приложение было оформлено при помощи “Bootstrap 4”. Дизайн строгий, минималистичный. Все элементы пользовательского интерфейса имеют адаптивный дизайн (способны подстраиваться под изменение размера на экране). Для удобства пользователей где это возможно применяются выпадающие списки, при вводе данных содержание некоторых из них изменяется в зависимости от выбора в вышестоящем списке. Такой подход позволяет сократить количество элементов списка облегчая работу с ним. Еще есть такие элементы как постраничный вывод, поиск и сортировка в прямом и обратном порядке.

При вводе данных применяется валидация ввода как на стороне клиента, так и на стороне сервера.

Защита приложения реализована при помощи встроенной системы ASP.NET Identity. Данные пользователей хранятся в базе данных на сервере. Пароли пользователей в базе данных хранятся в зашифрованном виде. Доступ к работе с приложением разрешен только авторизированным пользователям. Такие операции как удаление данных или изменение важных данных разрешены только некоторым пользователям.

Если локальная судовая сеть имеет постоянный выход в интернет, то можно реализовать двухфакторную идентификацию по e-mail или с использованием аккаунтов социальных сетей, в ASP.NET Identity есть поддержка этих опций. Доработка приложения не займет много времени и больших дополнительных затрат.

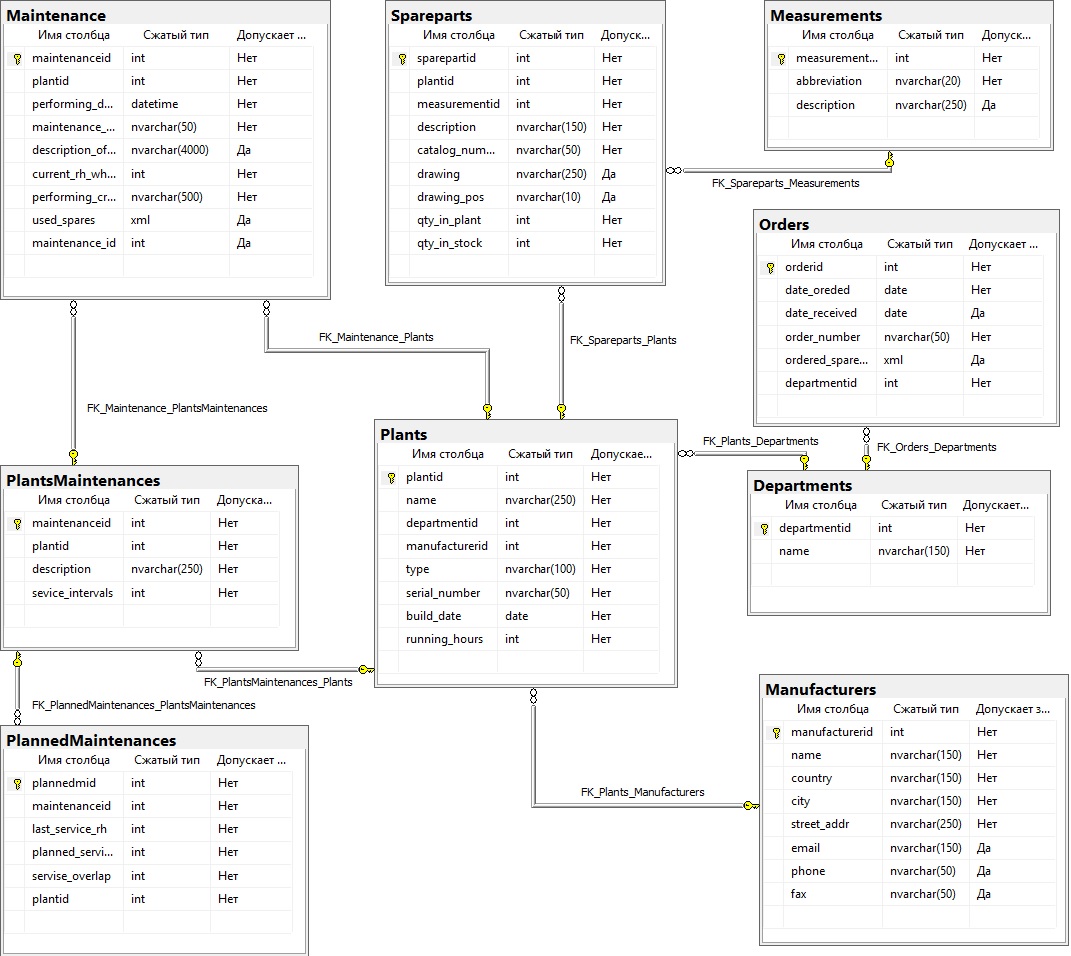
Для выявления возможных ошибок необходимо протестировать приложение в реальных условиях. Также в ходе работы с приложением необходимо изучить пожелания и замечания пользователей и в дальнейшем доработать приложение учитывая их.

# Список использованной литературы

* Методические материалы Компьютерной Академии «Шаг»;
* Интернет ресурс [https://msdn.microsoft.com](https://msdn.microsoft.com/);
* Интернет ресурс <http://www.csharp-examples.net/>;
* Интернет ресурс <https://www.codeproject.com/>;
* Интернет ресурс <https://professorweb.ru/>;
* Интернет ресурс <https://ru.wikipedia.org/>;
* Интернет ресурс <https://metanit.com/>;
* Интернет ресурс <https://getbootstrap.com/docs/4.0/components/>;
* Интернет ресурс <https://www.asp.net/>;
* Интернет ресурс <https://www.c-sharpcorner.com/>;
* Интернет ресурс <https://www.w3schools.com/>;
* Интернет ресурс <https://html5book.ru/>;
* Интернет ресурс <http://stackoverflow.com/>;
* Интернет — форум: <http://www.cyberforum.ru/>;

# Приложения

## Приложение №1



## Приложение №2

using System;

using System.ComponentModel.DataAnnotations;

using System.Web.Mvc;

namespace SparepartsManagementSystem.Models

{

public class DepartmentsMetadata

{

[Key]

public int departmentid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Department")]

public string name { get; set; }

}

public class MaintenanceMetadata

{

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Current RH on job complete")]

public int current\_rh\_when\_done { get; set; }

[StringLength(4000, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Description of work")]

[DataType(DataType.MultilineText)]

public string description\_of\_work { get; set; }

[Display(Name = "Maintenance")]

public int maintenance\_id { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Type of maintenance")]

public string maintenance\_type { get; set; }

[Key]

public int maintenanceid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(500, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Performing crew")]

[DataType(DataType.MultilineText)]

public string performing\_crew { get; set; }

[Required(ErrorMessage = "This field is required!")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

[Display(Name = "Performing date")]

public DateTime performing\_date { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Plant")]

public int plantid { get; set; }

[Display(Name = "Used spare parts")]

[AllowHtml]

public string used\_spares { get; set; }

}

public class ManufacturersMetadata

{

[Required(ErrorMessage = "This field is required!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "City")]

public string city { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Country")]

public string country { get; set; }

[DataType(DataType.EmailAddress, ErrorMessage = "Please input valid e-mail address!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "E-mail")]

public string email { get; set; }

[DataType(DataType.PhoneNumber, ErrorMessage = "Please input valid phone number!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Fax")]

public string fax { get; set; }

[Key]

public int manufacturerid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Manufacturer")]

public string name { get; set; }

[DataType(DataType.PhoneNumber, ErrorMessage = "Please input valid phone number!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Phone")]

public string phone { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(250, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Street address")]

public string street\_addr { get; set; }

}

public class MeasurementsMetadata

{

[Required(ErrorMessage = "This field is required!")]

[StringLength(20, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Abbreviation")]

public string abbreviation { get; set; }

[StringLength(250, ErrorMessage = "You reached maximum string length!")]

[DataType(DataType.MultilineText)]

[Display(Name = "Description")]

public string description { get; set; }

[Key]

public int measurementsid { get; set; }

}

public class OrdersMetadata

{

[Required(ErrorMessage = "This field is required!")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

[Display(Name = "Order date")]

public DateTime date\_oreded { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

[Display(Name = "Received date")]

public Nullable<System.DateTime> date\_received { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Department")]

public int departmentid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Order number")]

public string order\_number { get; set; }

[Display(Name = "Ordered spare parts")]

[AllowHtml]

public string ordered\_spareparts { get; set; }

[Key]

public int orderid { get; set; }

}

public class PlannedMaintenancesMetadata

{

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Last service RH")]

public int last\_service\_rh { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Maintenance")]

public int maintenanceid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Planned service at RH")]

public int planned\_service\_rh { get; set; }

[Key]

public int plannedmid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Plant")]

public int plantid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Service overlap")]

public int servise\_overlap { get; set; }

}

public class PlantsMaintenancesMetadata

{

[Required(ErrorMessage = "This field is required!")]

[StringLength(250, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Description")]

[DataType(DataType.MultilineText)]

public string description { get; set; }

[Key]

public int maintenanceid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Plant")]

public int plantid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Service intervals")]

public int sevice\_intervals { get; set; }

}

public class PlantsMetadata

{

[Required(ErrorMessage = "This field is required!")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

[Display(Name = "Build date")]

public DateTime build\_date { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Department")]

public int departmentid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Manufacturer")]

public int manufacturerid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(250, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Plant")]

public string name { get; set; }

[Key]

public int plantid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Running hours (RH)")]

public int running\_hours { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Serial number")]

public string serial\_number { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(100, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Type")]

public string type { get; set; }

}

public class SparepartsMetadata

{

[Required(ErrorMessage = "This field is required!")]

[StringLength(50, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Catalog number")]

public string catalog\_number { get; set; }

[Required(ErrorMessage = "This field is required!")]

[StringLength(150, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Description")]

[DataType(DataType.MultilineText)]

public string description { get; set; }

[StringLength(250, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Drawing")]

public string drawing { get; set; }

[StringLength(10, ErrorMessage = "You reached maximum string length!")]

[Display(Name = "Drawing position")]

public string drawing\_pos { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Measurement")]

public int measurementid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Plant")]

public int plantid { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Quantity in plant")]

public int qty\_in\_plant { get; set; }

[Required(ErrorMessage = "This field is required!")]

[RegularExpression("^\\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Quantity in stock")]

public int qty\_in\_stock { get; set; }

[Key]

public int sparepartid { get; set; }

}

}

## Приложение №3

using System.ComponentModel.DataAnnotations;

namespace SparepartsManagementSystem.Models

{

[MetadataType(typeof(DepartmentsMetadata))]

public partial class Departments

{

}

[MetadataType(typeof(MaintenanceMetadata))]

public partial class Maintenance

{

}

[MetadataType(typeof(ManufacturersMetadata))]

public partial class Manufacturers

{

}

[MetadataType(typeof(MeasurementsMetadata))]

public partial class Measurements

{

}

[MetadataType(typeof(OrdersMetadata))]

public partial class Orders

{

}

[MetadataType(typeof(PlannedMaintenancesMetadata))]

public partial class PlannedMaintenances

{

}

[MetadataType(typeof(PlantsMetadata))]

public partial class Plants

{

}

[MetadataType(typeof(PlantsMaintenancesMetadata))]

public partial class PlantsMaintenances

{

}

[MetadataType(typeof(SparepartsMetadata))]

public partial class Spareparts

{

public string ExtendedInfo => this.description + "; Cat No( " + this.catalog\_number + " )";

}

}

## Приложение №4

using SparepartsManagementSystem.Models;

using System;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using X.PagedList;

namespace SparepartsManagementSystem.Controllers

{

public class SparepartsController : Controller

{

private readonly SparepartsMSEntities db = new SparepartsMSEntities();

// GET: Spareparts/Create

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

public ActionResult Create()

{

ViewBag.IsActive = "Spareparts";

ViewBag.measurementid = new SelectList(db.Measurements, "measurementsid", "abbreviation");

ViewBag.plantid = new SelectList(db.Plants, "plantid", "name");

return View();

}

// POST: Spareparts/Create Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include =

"sparepartid,plantid,measurementid,description,catalog\_number,drawing,drawing\_pos,qty\_in\_plant,qty\_in\_stock")]

Spareparts spareparts, HttpPostedFileBase upload)

{

if (ModelState.IsValid)

{

if (upload != null && upload.ContentLength > 0)

try

{

var path = Path.Combine(Server.MapPath("~/Content/Images"),

Path.GetFileName(upload.FileName));

upload.SaveAs(path);

spareparts.drawing = @"~\Content\Images\" + Path.GetFileName(upload.FileName);

ViewBag.Message = "File uploaded successfully";

}

catch (Exception ex)

{

ViewBag.Message = "ERROR:" + ex.Message;

}

db.Spareparts.Add(spareparts);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.IsActive = "Spareparts";

ViewBag.measurementid =

new SelectList(db.Measurements, "measurementsid", "abbreviation", spareparts.measurementid);

ViewBag.plantid = new SelectList(db.Plants, "plantid", "name", spareparts.plantid);

return View(spareparts);

}

// GET: Spareparts/Delete/5

[Authorize(Users = "admin, Master")]

public ActionResult Delete(int? id, int page)

{

ViewBag.IsActive = "Spareparts";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var spareparts = db.Spareparts.Find(id);

if (spareparts == null) return HttpNotFound();

ViewBag.page = page;

return View(spareparts);

}

// POST: Spareparts/Delete/5

[HttpPost]

[ActionName("Delete")]

[Authorize(Users = "admin, Master")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id, int page)

{

ViewBag.IsActive = "Spareparts";

var spareparts = db.Spareparts.Find(id);

if (spareparts != null) db.Spareparts.Remove(spareparts);

db.SaveChanges();

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

}

// GET: Spareparts/Details/5

[Authorize]

public ActionResult Details(int? id, int page, string sortColumn, string searchText, string searchField,

string currentSortOrder)

{

ViewBag.IsActive = "Spareparts";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var spareparts = db.Spareparts.Find(id);

if (spareparts == null) return HttpNotFound();

ViewBag.backUrl = Url.Action("Index", new { page, sortColumn, currentSortOrder, searchText, searchField });

ViewBag.page = page;

return View(spareparts);

}

[Authorize]

public ActionResult Drawing(string drawingUrl, int page, string sortColumn, string searchText,

string searchField, string currentSortOrder)

{

ViewBag.IsActive = "Spareparts";

ViewBag.drawingUrl = drawingUrl;

ViewBag.backUrl = Url.Action("Index", new { page, sortColumn, currentSortOrder, searchText, searchField });

return View();

}

// GET: Spareparts/Edit/5

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

public ActionResult Edit(int? id, int page, string sortColumn, string searchText, string searchField,

string currentSortOrder, string backUrl)

{

ViewBag.IsActive = "Spareparts";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var spareparts = db.Spareparts.Find(id);

if (spareparts == null) return HttpNotFound();

ViewBag.measurementid =

new SelectList(db.Measurements, "measurementsid", "abbreviation", spareparts.measurementid);

ViewBag.plantid = new SelectList(db.Plants, "plantid", "name", spareparts.plantid);

ViewBag.backUrl =

backUrl ?? Url.Action("Index", new { page, sortColumn, currentSortOrder, searchText, searchField });

return View(spareparts);

}

// POST: Spareparts/Edit/5 Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include =

"sparepartid,plantid,measurementid,description,catalog\_number,drawing,drawing\_pos,qty\_in\_plant,qty\_in\_stock")]

Spareparts spareparts, HttpPostedFileBase upload, string backUrl)

{

if (ModelState.IsValid)

{

if (upload != null && upload.ContentLength > 0)

try

{

var path = Path.Combine(Server.MapPath("~/Content/Images"),

Path.GetFileName(upload.FileName));

upload.SaveAs(path);

spareparts.drawing = @"~\Content\Images\" + Path.GetFileName(upload.FileName);

ViewBag.Message = "File uploaded successfully";

}

catch (Exception ex)

{

ViewBag.Message = "ERROR:" + ex.Message;

}

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

db.SaveChanges();

return Redirect(backUrl);

}

ViewBag.IsActive = "Spareparts";

ViewBag.measurementid =

new SelectList(db.Measurements, "measurementsid", "abbreviation", spareparts.measurementid);

ViewBag.plantid = new SelectList(db.Plants, "plantid", "name", spareparts.plantid);

ViewBag.backUrl = backUrl;

return View(spareparts);

}

// GET: Spareparts

[Authorize]

public ActionResult Index(int? page, string sortColumn, string searchText, string searchField,

string currentSortOrder)

{

ViewBag.IsActive = "Spareparts";

var pageNumber = page ?? 1;

ViewBag.currentSortOrder = currentSortOrder;

ViewBag.sortColumn = sortColumn = string.IsNullOrEmpty(sortColumn) ? "sparepartid" : sortColumn;

IQueryable<Spareparts> modelList = null;

switch (sortColumn)

{

case "sparepartid":

{

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderBy(s => s.sparepartid);

break;

}

case "plantid":

{

if (!string.IsNullOrEmpty(currentSortOrder) && currentSortOrder.Equals("desc"))

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderByDescending(s => s.plantid);

else

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderBy(s => s.plantid);

break;

}

case "description":

{

if (!string.IsNullOrEmpty(currentSortOrder) && currentSortOrder.Equals("desc"))

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderByDescending(s => s.description);

else

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderBy(s => s.description);

break;

}

case "catalog\_number":

{

if (!string.IsNullOrEmpty(currentSortOrder) && currentSortOrder.Equals("desc"))

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderByDescending(s => s.catalog\_number);

else

modelList = db.Spareparts.Include(s => s.Measurements).Include(s => s.Plants)

.OrderBy(s => s.catalog\_number);

break;

}

}

if (!string.IsNullOrEmpty(searchField) && !searchField.Equals("none"))

{

switch (searchField)

{

case "catalog\_number":

{

modelList = modelList.Where(x => x.catalog\_number.Contains(searchText));

break;

}

case "description":

{

modelList = modelList.Where(x => x.description.Contains(searchText));

break;

}

case "plantid":

{

modelList = modelList.Where(x => x.Plants.name.Contains(searchText));

break;

}

}

ViewBag.searchText = searchText;

ViewBag.searchField = searchField;

return View(modelList.ToPagedList(pageNumber, 10));

}

ViewBag.searchField = "none";

return View(modelList.ToPagedList(pageNumber, 10));

}

protected override void Dispose(bool disposing)

{

if (disposing) db.Dispose();

base.Dispose(disposing);

}

}

}

@model X.PagedList.IPagedList<SparepartsManagementSystem.Models.Spareparts>

@using X.PagedList.Mvc;

@using X.PagedList.Mvc.Bootstrap4

@{

ViewBag.Title = "List of spare parts";

}

<h2>@ViewBag.Title</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<form class="form-inline" method="post">

<label for="searchText" class="col-form-label col-1">Search</label>

<input name="searchText" id="searchText" type="text" placeholder="Search for..." class="form-control col-4" />

<label for="searchField" class="col-form-label col-2">Search field</label>

<select name="searchField" id="searchField" class="form-control col-3">

<option value="none" selected="selected">None</option>

<option value="plantid">Plant</option>

<option value="description">Description</option>

<option value="catalog\_number">Catalog number</option>

</select>

<input type="hidden" value="@ViewBag.currentSortOrder" name="currentSortOrder" id="currentSortOrder" />

<input type="hidden" value="@ViewBag.sortColumn" name="sortColumn" id="sortColumn" />

<input type="hidden" value="1" name="page" id="page" />

<button name="sbm\_btn" id="sbm\_btn" type="submit" class="btn btn-outline-info col-1">Search</button>

</form>

<br />

@{

var decr\_class = "";

var catalog\_class = "";

var plant\_class = "";

switch (ViewBag.sortColumn)

{

case "description":

{

decr\_class = ViewBag.currentSortOrder == "asc" ? "fa fa-sort-alpha-asc" : "fa fa-sort-alpha-desc";

break;

}

case "catalog\_number":

{

catalog\_class = ViewBag.currentSortOrder == "asc" ? "fa fa-sort-alpha-asc" : "fa fa-sort-alpha-desc";

break;

}

case "plantid":

{

plant\_class = ViewBag.currentSortOrder == "asc" ? "fa fa-sort-alpha-asc" : "fa fa-sort-alpha-desc";

break;

}

}

}

<div class="table-responsive">

<table class="table table-striped">

<thead>

<tr>

<th>

@Html.ActionLink("Description", "Index", new { sortColumn = "description", currentSortOrder = ViewBag.currentSortOrder == "asc" ? "desc" : "asc", page = Model.PageNumber, ViewBag.searchText, ViewBag.searchField })<span id="dsc\_span" class="@decr\_class"></span>

</th>

<th>

@Html.ActionLink("Catalog number", "Index", new { sortColumn = "catalog\_number", currentSortOrder = ViewBag.currentSortOrder == "asc" ? "desc" : "asc", page = Model.PageNumber, ViewBag.searchText, ViewBag.searchField })<span id="cat\_span" class="@catalog\_class"></span>

</th>

<th>

@Html.DisplayNameFor(model => model.GetEnumerator().Current.drawing)

</th>

<th>

@Html.DisplayNameFor(model => model.GetEnumerator().Current.drawing\_pos)

</th>

<th>

@Html.DisplayNameFor(model => model.GetEnumerator().Current.qty\_in\_plant)

</th>

<th>

@Html.DisplayNameFor(model => model.GetEnumerator().Current.qty\_in\_stock)

</th>

<th>

@Html.DisplayNameFor(model => model.GetEnumerator().Current.Measurements.abbreviation)

</th>

<th>

@Html.ActionLink("Plant", "Index", new { sortColumn = "plantid", currentSortOrder = ViewBag.currentSortOrder == "asc" ? "desc" : "asc", page = Model.PageNumber, ViewBag.searchText, ViewBag.searchField })<span id="pl\_span" class="@plant\_class"></span>

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.description)

</td>

<td>

@Html.DisplayFor(modelItem => item.catalog\_number)

</td>

<td>

@if (item.drawing != null)

{

<a href="@Url.Action("Drawing", "Spareparts", new {drawingUrl = Url.Content(item.drawing), page = Model.PageNumber, ViewBag.searchText, ViewBag.searchField, ViewBag.sortColumn, ViewBag.currentSortOrder})"><img src="@Url.Content(item.drawing)" class="img-thumbnail" alt="@item.ExtendedInfo" /></a>

}

else

{

<label class="text-info">Drawing not available.</label>

}

</td>

<td>

@Html.DisplayFor(modelItem => item.drawing\_pos)

</td>

<td>

@Html.DisplayFor(modelItem => item.qty\_in\_plant)

</td>

<td>

@Html.DisplayFor(modelItem => item.qty\_in\_stock)

</td>

<td>

@Html.DisplayFor(modelItem => item.Measurements.abbreviation)

</td>

<td>

@Html.DisplayFor(modelItem => item.Plants.name)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.sparepartid, page = Model.PageNumber, ViewBag.sortColumn, ViewBag.currentSortOrder, ViewBag.searchField, ViewBag.searchText }) |

@Html.ActionLink("Details", "Details", new { id = item.sparepartid, page = Model.PageNumber, ViewBag.sortColumn, ViewBag.currentSortOrder, ViewBag.searchField, ViewBag.searchText }) |

@Html.ActionLink("Delete", "Delete", new { id = item.sparepartid, page = Model.PageNumber, ViewBag.sortColumn, ViewBag.currentSortOrder, ViewBag.searchField, ViewBag.searchText })

</td>

</tr>

}

</tbody>

</table>

</div>

<br />

Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCount

<br />

@Html.PagedListPager(Model, page => Url.Action("Index", new { page, ViewBag.sortColumn, ViewBag.currentSortOrder, ViewBag.searchField, ViewBag.searchText }), Bootstrap4PagedListRenderOptions.ClassicPlusFirstAndLast)

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

<script type="text/javascript">

$(document).ready(function() {

$(function() {

$("#searchField").val('@ViewBag.searchField');

$("#searchText").val('@ViewBag.searchText');

if ('@ViewBag.searchField' === 'none') {

$("#searchText").prop("disabled", true);

$("#searchText").val(null);

$("#sbm\_btn").prop("disabled", true);

console.log("Fields disabled!");

}

});

$("#searchField").change(function() {

var key = $("#searchField :selected").text();

switch (key) {

case "None":

{

$("#searchText").prop("disabled", true);

$("#searchText").val(null);

$("#sbm\_btn").prop("disabled", true);

console.log("Fields disabled!");

window.location.href = '/Spareparts/Index';

break;

}

default:

{

$("#searchText").prop("disabled", false);

$("#searchText").val('@ViewBag.searchText');

$("#sbm\_btn").prop("disabled", false);

console.log("Fields enabled!");

break;

}

}

});

});

$(function(sortColumn = "@ViewBag.sortColumn") {

switch (sortColumn) {

case "description":

{

$("#dsc\_span").show();

$("#cat\_span").hide();

$("#pl\_span").hide();

break;

}

case "catalog\_number":

{

$("#cat\_span").show();

$("#dsc\_span").hide();

$("#pl\_span").hide();

break;

}

case "plantid":

{

$("#pl\_span").show();

$("#cat\_span").hide();

$("#dsc\_span").hide();

break;

}

case "sparepartid":

{

$("#pl\_span").hide();

$("#cat\_span").hide();

$("#dsc\_span").hide();

break;

}

}

});

</script>

}

## Приложение №5

using SparepartsManagementSystem.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Web.Mvc;

using System.Xml;

namespace SparepartsManagementSystem.Controllers

{

public class UsedSparepartController : Controller

{

private readonly SparepartsMSEntities \_db = new SparepartsMSEntities();

public ActionResult Create(int plantid, int id)

{

ViewBag.plantid = plantid;

ViewBag.id = id;

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(pl => pl.plantid == plantid), "sparepartid",

"ExtendedInfo");

var model = new UsedSparepartModel();

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(int plantid, int id, UsedSparepartModel model)

{

if (ModelState.IsValid)

{

XmlDocument doc;

var maintenance = \_db.Maintenance.Find(id);

if (maintenance?.used\_spares != null)

{

doc = new XmlDocument();

doc.LoadXml(maintenance.used\_spares);

model.AppendModelToXmlDocument(doc);

doc.Save(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

}

else

{

doc = new XmlDocument();

var rootNode = doc.CreateElement("UsedSpareparts");

doc.AppendChild(rootNode);

model.AppendModelToXmlDocument(doc);

doc.Save(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

}

try

{

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Maintenance] SET [used\_spares] = @p0 WHERE maintenanceid = @p1",

doc.InnerXml, id);

\_db.SaveChanges();

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

return RedirectToAction("Index", new { id, plantid });

}

ViewBag.plantid = plantid;

ViewBag.id = id;

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(pl => pl.plantid == plantid), "sparepartid",

"ExtendedInfo");

return View(model);

}

public ActionResult Delete(int id, int sparepartid, int plantid)

{

try

{

ViewBag.id = id;

var doc = new XmlDocument();

doc.Load(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

var node = doc.SelectSingleNode($"UsedSpareparts/sparepart[sparepartId = '{sparepartid}']");

node.ParentNode.RemoveChild(node);

var root = doc.SelectSingleNode("UsedSpareparts");

if (!root.HasChildNodes)

{

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Maintenance] SET [used\_spares] = @p0 WHERE maintenanceid = @p1",

null, id);

\_db.SaveChanges();

System.IO.File.Delete(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

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

}

doc.Save(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Maintenance] SET [used\_spares] = @p0 WHERE maintenanceid = @p1",

doc.InnerXml, id);

\_db.SaveChanges();

return RedirectToAction("Index", new { id, plantid });

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

}

public ActionResult Edit(int plantid, int id, int sparepartid)

{

try

{

ViewBag.id = id;

ViewBag.plantid = plantid;

ViewBag.oldid = sparepartid;

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(pl => pl.plantid == plantid), "sparepartid",

"ExtendedInfo");

var doc = new XmlDocument();

doc.Load(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

var node = doc.SelectSingleNode($"UsedSpareparts/sparepart[sparepartId = '{sparepartid}']");

var model = new UsedSparepartModel(int.Parse(node["sparepartId"].InnerText),

int.Parse(node["quantity"].InnerText), node["reason"].InnerText);

return View(model);

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int plantid, int id, int oldid, UsedSparepartModel model)

{

if (ModelState.IsValid)

{

try

{

var doc = new XmlDocument();

doc.Load(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

var node = doc.SelectSingleNode($"UsedSpareparts/sparepart[sparepartId = '{oldid}']");

node["sparepartId"].InnerText = model.sparepartid.ToString();

node["quantity"].InnerText = model.Quantity.ToString();

node["reason"].InnerText = model.Reason;

doc.Save(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Maintenance] SET [used\_spares] = @p0 WHERE maintenanceid = @p1",

doc.InnerXml, id);

\_db.SaveChanges();

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

return RedirectToAction("Index", new { id, plantid });

}

ViewBag.plantid = plantid;

ViewBag.oldid = oldid;

ViewBag.id = id;

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(pl => pl.plantid == plantid), "sparepartid",

"ExtendedInfo");

return View(model);

}

// GET: UsedSparepart

public ActionResult Index(int id, int plantid)

{

var maintenance = \_db.Maintenance.Find(id);

if (maintenance?.used\_spares != null)

try

{

ViewBag.plantid = plantid;

ViewBag.id = id;

var doc = new XmlDocument();

doc.LoadXml(maintenance.used\_spares);

doc.Save(Server.MapPath($"~/App\_Data/{id}usedspares.xml"));

var models = new List<UsedSparepartModel>();

foreach (XmlNode node in doc.SelectNodes("UsedSpareparts/sparepart"))

{

var usedSparepartModel = new UsedSparepartModel(int.Parse(node["sparepartId"].InnerText),

int.Parse(node["quantity"].InnerText), node["reason"].InnerText);

models.Add(usedSparepartModel);

}

return View(models);

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

protected override void Dispose(bool disposing)

{

if (disposing) \_db.Dispose();

base.Dispose(disposing);

}

}

}

using System;

using System.ComponentModel.DataAnnotations;

using System.Runtime.Serialization;

using System.Xml;

namespace SparepartsManagementSystem.Models

{

[Serializable]

[DataContract]

public class UsedSparepartModel

{

public UsedSparepartModel()

{

this.sparepartid = 0;

this.Quantity = 0;

this.Reason = "";

}

public UsedSparepartModel(int id, int quantity, string reason)

{

this.sparepartid = id;

this.Quantity = quantity;

this.Reason = reason;

}

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[RegularExpression(@"^\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Quantity")]

public int Quantity { get; set; }

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[StringLength(4000, ErrorMessage = "You reach maximum field length!")]

[DataType(DataType.MultilineText)]

[Display(Name = "Reason")]

public string Reason { get; set; }

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[RegularExpression(@"^\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Spare part ID")]

public int sparepartid { get; set; }

public void AppendModelToXmlDocument(XmlDocument doc)

{

try

{

XmlNode rootNode = doc.GetElementsByTagName("UsedSpareparts")[0];

XmlNode nodeSparePart = doc.CreateElement("sparepart");

XmlNode nodeSparepartId = doc.CreateElement("sparepartId");

XmlNode nodeQuantity = doc.CreateElement("quantity");

XmlNode nodeReason = doc.CreateElement("reason");

nodeSparepartId.InnerText = sparepartid.ToString();

nodeQuantity.InnerText = Quantity.ToString();

nodeReason.InnerText = Reason;

nodeSparePart.AppendChild(nodeSparepartId);

nodeSparePart.AppendChild(nodeQuantity);

nodeSparePart.AppendChild(nodeReason);

rootNode.AppendChild(nodeSparePart);

}

catch (Exception e)

{

Console.Error.Write(e.Message);

throw;

}

}

}

}

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using SparepartsManagementSystem.Models;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

namespace SparepartsManagementSystem.Controllers

{

public class MaintenancesController : Controller

{

private readonly SparepartsMSEntities \_db = new SparepartsMSEntities();

private ApplicationUserManager \_userManager;

private ApplicationUserManager UserManager

{

get => \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

set => \_userManager = value;

}

// GET: Maintenances/Create

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

public ActionResult Create()

{

ViewBag.IsActive = "Maintenances";

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name");

return View();

}

// POST: Maintenances/Create Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include =

"maintenanceid,plantid,performing\_date,maintenance\_type,description\_of\_work,current\_rh\_when\_done,performing\_crew,used\_spares,maintenance\_id")]

Maintenance maintenance)

{

if (ModelState.IsValid)

{

\_db.Maintenance.Add(maintenance);

\_db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.IsActive = "Maintenances";

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name", maintenance.plantid);

return View(maintenance);

}

// GET: Maintenances/Delete/5

[Authorize(Users = "admin, Master")]

public ActionResult Delete(int? id)

{

ViewBag.IsActive = "Maintenances";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var maintenance = \_db.Maintenance.Find(id);

if (maintenance == null) return HttpNotFound();

return View(maintenance);

}

// POST: Maintenances/Delete/5

[HttpPost]

[ActionName("Delete")]

[Authorize(Users = "admin, Master")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

ViewBag.IsActive = "Maintenances";

var maintenance = \_db.Maintenance.Find(id);

\_db.Maintenance.Remove(maintenance);

\_db.SaveChanges();

return RedirectToAction("Index");

}

// GET: Maintenances/Details/5

[Authorize]

public ActionResult Details(int? id)

{

ViewBag.IsActive = "Maintenances";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var maintenance = \_db.Maintenance.Find(id);

if (maintenance == null) return HttpNotFound();

return View(maintenance);

}

// GET: Maintenances/Edit/5

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

public ActionResult Edit(int? id)

{

ViewBag.IsActive = "Maintenances";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var maintenance = \_db.Maintenance.Find(id);

if (maintenance == null) return HttpNotFound();

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name", maintenance.plantid);

ViewBag.maintenance\_id = new SelectList(\_db.PlantsMaintenances, "maintenanceid", "description",

maintenance.maintenance\_id);

return View(maintenance);

}

// POST: Maintenances/Edit/5 Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include =

"maintenanceid,plantid,performing\_date,maintenance\_type,description\_of\_work,current\_rh\_when\_done,performing\_crew,used\_spares,maintenance\_id")]

Maintenance maintenance)

{

if (ModelState.IsValid)

{

\_db.Entry(maintenance).State = EntityState.Modified;

\_db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.IsActive = "Maintenances";

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name", maintenance.plantid);

ViewBag.maintenance\_id = new SelectList(\_db.PlantsMaintenances, "maintenanceid", "description",

maintenance.maintenance\_id);

return View(maintenance);

}

[HttpPost]

public JsonResult GetMaintenances(int plantid)

{

return Json(new SelectList(\_db.PlantsMaintenances.Where(x => x.plantid == plantid), "maintenanceid",

"description", JsonRequestBehavior.AllowGet));

}

// GET: Maintenances

[Authorize]

public ActionResult Index()

{

ViewBag.IsActive = "Maintenances";

var user = UserManager.FindById(User.Identity.GetUserId());

var maintenance = \_db.Maintenance.Include(m => m.Plants).Include(p => p.PlantsMaintenances).Where(p => p.Plants.Departments.name == user.department);

return View(maintenance.ToList());

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

if (\_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

\_db?.Dispose();

}

base.Dispose(disposing);

}

}

}

@model SparepartsManagementSystem.Models.Maintenance

@{

ViewBag.Title = "Create";

}

<h2 class="h2">Create</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<h4 class="h4">Maintenance</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.plantid, "Plant", htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("plantid", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.plantid, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.performing\_date, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.performing\_date, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.performing\_date, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.maintenance\_type, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.maintenance\_type, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.maintenance\_type, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.description\_of\_work, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.description\_of\_work, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.description\_of\_work, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.current\_rh\_when\_done, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.current\_rh\_when\_done, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.current\_rh\_when\_done, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.performing\_crew, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.performing\_crew, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.performing\_crew, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.used\_spares, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.used\_spares, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.used\_spares, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.maintenance\_id, "Planned maintenance", htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("maintenance\_id", new SelectList(string.Empty, "maintenanceid", "description"), "Select planned maintenance (if applicable)", htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.maintenance\_id, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="offset-md-2 col-md-10">

<input type="submit" value="Create" class="btn btn-outline-success" />

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

<script type="text/javascript">

$(document).ready(function () {

$("#plantid").change(function() {

$("#maintenance\_id").empty();

$.ajax({

type: 'POST',

url: '@Url.Action("GetMaintenances")',

dataType: 'json',

data: { plantid: $("#plantid").val() },

success: function (spareparts) {

var newOption = $('<option value=' + null + ' >Select planned maintenance (if applicable)</option>');

$("#maintenance\_id").append(newOption);

$.each(spareparts,

function(i, item) {

$("#maintenance\_id").append('<option value="' +

item.Value +

'">' +

item.Text +

'</option>');

});

},

error: function(ex) {

alert('Failed to retrieve maintenances.' + ex);

}

});

return false;

});

});

</script>

}

## Приложение №6

using System;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Runtime.Serialization;

using System.Xml;

namespace SparepartsManagementSystem.Models

{

[Serializable]

[DataContract]

public class OrderedSparepartModel

{

public OrderedSparepartModel()

{

Plantid = 0;

Sparepartid = 0;

Quantity = 0;

PlantName = "";

ExtendedInfo = "";

}

public OrderedSparepartModel(int plantid, int sparepartid, int quantity, string plantName, string extendedInfo)

{

Plantid = plantid;

Sparepartid = sparepartid;

Quantity = quantity;

PlantName = plantName;

ExtendedInfo = extendedInfo;

}

[DataMember] public string ExtendedInfo { get; set; }

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Plant")]

public int Plantid { get; set; }

[DataMember] public string PlantName { get; set; }

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[RegularExpression(@"^\d+$", ErrorMessage = "Allowed digits only!")]

[Display(Name = "Quantity")]

public int Quantity { get; set; }

[DataMember]

[Required(ErrorMessage = "This field is required!")]

[Display(Name = "Spare part")]

public int Sparepartid { get; set; }

public void AppendModelToXmlDocument(XmlDocument doc, IQueryable<Plants> plants)

{

try

{

var exist = false;

var rootNode = doc.SelectSingleNode("OrderedSpareparts/Plants");

XmlNode plantNode;

if (rootNode?.SelectSingleNode($"plantid[@id='{Plantid}']") != null)

{

plantNode = rootNode.SelectSingleNode($"plantid[@id='{Plantid}']");

exist = true;

}

else

{

plantNode = doc.CreateElement("plantid");

var idAttribute = doc.CreateAttribute("id");

idAttribute.Value = Plantid.ToString();

var manufAttribute = doc.CreateAttribute("manufacturer");

var plant = plants.Where(p => p.plantid == Plantid);

manufAttribute.Value = plant.Select(p => p.Manufacturers.name).FirstOrDefault();

var manufEmailAttribute = doc.CreateAttribute("manufacturerEmail");

manufEmailAttribute.Value = plant.Select(p => p.Manufacturers.email).FirstOrDefault();

var manufFaxAttribute = doc.CreateAttribute("manufacturerFax");

manufFaxAttribute.Value = plant.Select(p => p.Manufacturers.fax).FirstOrDefault();

var serialnumberAttribute = doc.CreateAttribute("serialNumber");

serialnumberAttribute.Value = plant.Select(p => p.serial\_number).FirstOrDefault();

plantNode.InnerText = PlantName;

plantNode.Attributes.Append(idAttribute);

plantNode.Attributes.Append(serialnumberAttribute);

plantNode.Attributes.Append(manufAttribute);

plantNode.Attributes.Append(manufFaxAttribute);

plantNode.Attributes.Append(manufEmailAttribute);

exist = false;

}

XmlNode sparepartNode = doc.CreateElement("sparepart");

var spr\_idAttribute = doc.CreateAttribute("id");

spr\_idAttribute.Value = Sparepartid.ToString();

var quantAttribute = doc.CreateAttribute("quantity");

quantAttribute.Value = Quantity.ToString();

sparepartNode.InnerText = ExtendedInfo;

sparepartNode.Attributes.Append(spr\_idAttribute);

sparepartNode.Attributes.Append(quantAttribute);

plantNode.AppendChild(sparepartNode);

if (!exist) rootNode.AppendChild(plantNode);

}

catch (Exception e)

{

Console.Error.Write(e.Message);

throw;

}

}

}

}

using SparepartsManagementSystem.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web.Mvc;

using System.Xml;

namespace SparepartsManagementSystem.Controllers

{

public class OrderedSparepartController : Controller

{

private readonly SparepartsMSEntities \_db = new SparepartsMSEntities();

public ActionResult Create(int orderid)

{

ViewBag.orderid = orderid;

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name");

var model = new OrderedSparepartModel();

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(OrderedSparepartModel model, int orderid)

{

if (ModelState.IsValid)

{

var plants = \_db.Plants.Include(p => p.Manufacturers);

XmlDocument doc;

var order = \_db.Orders.Find(orderid);

if (order?.ordered\_spareparts != null)

{

doc = new XmlDocument();

doc.LoadXml(order.ordered\_spareparts);

model.AppendModelToXmlDocument(doc, plants);

doc.Save(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

}

else

{

doc = new XmlDocument();

var rootNode = doc.CreateElement("OrderedSpareparts");

var plantsNode = doc.CreateElement("Plants");

rootNode.AppendChild(plantsNode);

doc.AppendChild(rootNode);

model.AppendModelToXmlDocument(doc, plants);

doc.Save(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

}

try

{

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Orders] SET [ordered\_spareparts] = @p0 WHERE orderid = @p1",

doc.InnerXml, orderid);

\_db.SaveChanges();

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

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

}

ViewBag.orderid = orderid;

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name");

return View(model);

}

public ActionResult Delete(int orderid, int sparepartid, int plantid)

{

try

{

var doc = new XmlDocument();

doc.Load(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

var plantNode = doc.SelectSingleNode($"OrderedSpareparts/Plants/plantid[@id = '{plantid}']");

var sparepartNode = plantNode.SelectSingleNode($"sparepart[@id = '{sparepartid}']");

sparepartNode.ParentNode.RemoveChild(sparepartNode);

var sparepartNodes =

plantNode.SelectNodes($"OrderedSpareparts/Plants/plantid[@id = '{plantid}']/sparepart");

if (sparepartNodes.Count == 0) plantNode.ParentNode.RemoveChild(plantNode);

var plantsNode = doc.SelectSingleNode("OrderedSpareparts/Plants");

if (!plantsNode.HasChildNodes)

{

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Orders] SET [ordered\_spareparts] = @p0 WHERE orderid = @p1",

null, orderid);

\_db.SaveChanges();

System.IO.File.Delete(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

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

}

doc.Save(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Orders] SET [ordered\_spareparts] = @p0 WHERE orderid = @p1",

doc.InnerXml, orderid);

\_db.SaveChanges();

ViewBag.orderid = orderid;

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

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

}

public ActionResult Edit(int orderid, int sparepartid, int plantid)

{

try

{

var doc = new XmlDocument();

doc.Load(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

var plantNode = doc.SelectSingleNode($"OrderedSpareparts/Plants/plantid[@id = '{plantid}']");

var sparepartNode = plantNode.SelectSingleNode($"sparepart[@id = '{sparepartid}']");

var model = new OrderedSparepartModel(int.Parse(plantNode.Attributes["id"].Value),

int.Parse(sparepartNode.Attributes["id"].Value),

int.Parse(sparepartNode.Attributes["quantity"].Value), plantNode.FirstChild.InnerText,

sparepartNode.InnerText);

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name", model.Plantid);

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(s => s.plantid == model.Plantid),

"sparepartid", "ExtendedInfo", model.Sparepartid);

ViewBag.orderid = orderid;

ViewBag.oldsparepartid = sparepartid;

ViewBag.oldplantid = plantid;

return View(model);

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int orderid, int oldsparepartid, int oldplantid, OrderedSparepartModel model)

{

if (ModelState.IsValid)

{

try

{

var plants = \_db.Plants.Include(p => p.Manufacturers);

var doc = new XmlDocument();

doc.Load(System.Web.HttpContext.Current.Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

if (oldplantid == model.Plantid)

{

var plantNode = doc.SelectSingleNode($"OrderedSpareparts/Plants/plantid[@id = '{oldplantid}']");

var sparepartNode = plantNode.SelectSingleNode($"sparepart[@id = '{oldsparepartid}']");

sparepartNode.Attributes["id"].Value = model.Sparepartid.ToString();

sparepartNode.Attributes["quantity"].Value = model.Quantity.ToString();

sparepartNode.InnerText = model.ExtendedInfo;

}

else

{

var plantNode = doc.SelectSingleNode($"OrderedSpareparts/Plants/plantid[@id = '{oldplantid}']");

var sparepartNode = plantNode.SelectSingleNode($"sparepart[@id = '{oldsparepartid}']");

sparepartNode.ParentNode.RemoveChild(sparepartNode);

if (!plantNode.HasChildNodes) plantNode.ParentNode.RemoveChild(plantNode);

model.AppendModelToXmlDocument(doc, plants);

}

doc.Save(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

\_db.Database.ExecuteSqlCommand(

"UPDATE [dbo].[Orders] SET [ordered\_spareparts] = @p0 WHERE orderid = @p1",

doc.InnerXml, orderid);

\_db.SaveChanges();

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

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

}

try

{

ViewBag.plantid = new SelectList(\_db.Plants, "plantid", "name", model.Plantid);

ViewBag.sparepartid = new SelectList(\_db.Spareparts.Where(s => s.plantid == model.Plantid),

"sparepartid", "ExtendedInfo", model.Sparepartid);

ViewBag.orderid = orderid;

ViewBag.oldsparepartid = oldsparepartid;

ViewBag.oldplantid = oldplantid;

return View(model);

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

}

[HttpPost]

public JsonResult GetSpareparts(int plantid)

{

return Json(new SelectList(\_db.Spareparts.Where(x => x.plantid == plantid), "sparepartid", "ExtendedInfo",

JsonRequestBehavior.AllowGet));

}

// GET: OrderedSparepart

public ActionResult Index(int orderid)

{

var order = \_db.Orders.Find(orderid);

if (order?.ordered\_spareparts != null)

try

{

var doc = new XmlDocument();

doc.LoadXml(order.ordered\_spareparts);

doc.Save(Server.MapPath($"~/App\_Data/{orderid}orderedspares.xml"));

var models = new List<OrderedSparepartModel>();

foreach (XmlNode node in doc.SelectNodes("OrderedSpareparts/Plants"))

foreach (XmlNode plNode in node.SelectNodes("plantid"))

{

var plantid = int.Parse(plNode.Attributes["id"].Value);

var plantName = plNode.FirstChild.InnerText;

foreach (XmlNode item in plNode.SelectNodes("sparepart"))

{

var orderedSparepartsModel = new OrderedSparepartModel(plantid,

int.Parse(item.Attributes["id"].Value), int.Parse(item.Attributes["quantity"].Value),

plantName, item.InnerText);

models.Add(orderedSparepartsModel);

}

}

ViewBag.orderid = orderid;

return View(models);

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

protected override void Dispose(bool disposing)

{

if (disposing) \_db.Dispose();

base.Dispose(disposing);

}

}

}

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using SelectPdf;

using SparepartsManagementSystem.Models;

using SparepartsManagementSystem.Properties;

using System;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Web;

using System.Web.Mvc;

using System.Xml;

using System.Xml.Xsl;

namespace SparepartsManagementSystem.Controllers

{

public class OrdersController : Controller

{

private readonly SparepartsMSEntities \_db = new SparepartsMSEntities();

private ApplicationUserManager \_userManager;

private ApplicationUserManager UserManager

{

get => \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

set => \_userManager = value;

}

// GET: Orders/Create

[Authorize]

public ActionResult Create()

{

ViewBag.IsActive = "Orders";

ViewBag.departmentid = new SelectList(\_db.Departments, "departmentid", "name");

return View();

}

// POST: Orders/Create Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "orderid,date\_oreded,date\_received,order\_number,ordered\_spareparts,departmentid")]

Orders orders)

{

if (ModelState.IsValid)

{

\_db.Orders.Add(orders);

\_db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.IsActive = "Orders";

ViewBag.departmentid = new SelectList(\_db.Departments, "departmentid", "name");

return View(orders);

}

// GET: Orders/Delete/5

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

public ActionResult Delete(int? id)

{

ViewBag.IsActive = "Orders";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var orders = \_db.Orders.Find(id);

if (orders == null) return HttpNotFound();

return View(orders);

}

// POST: Orders/Delete/5

[HttpPost]

[ActionName("Delete")]

[Authorize(Users = "admin, Master, Chief Officer, Chief Engineer")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

ViewBag.IsActive = "Orders";

var orders = \_db.Orders.Find(id);

\_db.Orders.Remove(orders);

\_db.SaveChanges();

return RedirectToAction("Index");

}

// GET: Orders/Details/5

[Authorize]

public ActionResult Details(int? id)

{

ViewBag.IsActive = "Orders";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var orders = \_db.Orders.Find(id);

if (orders == null) return HttpNotFound();

return View(orders);

}

// GET: Orders/Edit/5

[Authorize]

public ActionResult Edit(int? id)

{

ViewBag.IsActive = "Orders";

if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

var orders = \_db.Orders.Find(id);

if (orders == null) return HttpNotFound();

return View(orders);

}

// POST: Orders/Edit/5 Чтобы защититься от атак чрезмерной передачи данных, включите

// определенные свойства, для которых следует установить привязку. Дополнительные сведения

// см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "orderid,date\_oreded,date\_received,order\_number,ordered\_spareparts,departmentid")]

Orders orders)

{

if (ModelState.IsValid)

{

\_db.Entry(orders).State = EntityState.Modified;

\_db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.IsActive = "Orders";

return View(orders);

}

public FileResult GetPdf(int? id)

{

var path = $"~/App\_Data/{id}orderedspares.xml";

var order = \_db.Orders.Find(id);

if (order?.ordered\_spareparts != null)

{

try

{

var doc = new XmlDocument();

doc.LoadXml(order.ordered\_spareparts);

var companyNode = doc.CreateElement("companyInfo");

var orderNode = doc.CreateElement("orderInfo");

var vesselNode = doc.CreateElement("vesselInfo");

var rootNode = doc.SelectSingleNode("OrderedSpareparts");

var companyName = doc.CreateAttribute("companyName");

var departmentName = doc.CreateAttribute("departmentName");

departmentName.Value = order.Departments.name;

companyName.Value = Resources.companyName;

var vesselName = doc.CreateAttribute("vesselName");

vesselName.Value = Resources.vesselName;

var orderNumber = doc.CreateAttribute("orderNumber");

orderNumber.Value = order.order\_number;

companyNode.Attributes.Append(companyName);

vesselNode.Attributes.Append(vesselName);

orderNode.Attributes.Append(orderNumber);

orderNode.Attributes.Append(departmentName);

rootNode.AppendChild(companyNode);

rootNode.AppendChild(vesselNode);

rootNode.AppendChild(orderNode);

doc.Save(Server.MapPath(path));

var xslt = new XslCompiledTransform();

xslt.Load(Server.MapPath("~/App\_Data/transform.xsl"));

var xmlTextWriter = new XmlTextWriter(Server.MapPath($"~/App\_Data/{order.order\_number}.html"),

Encoding.UTF8)

{

Formatting = Formatting.Indented,

Indentation = 4

};

xslt.Transform(doc, xmlTextWriter);

xmlTextWriter.Close();

xmlTextWriter.Dispose();

var converter = new HtmlToPdf();

converter.Options.PdfPageSize = PdfPageSize.A4;

converter.Options.PdfPageOrientation = PdfPageOrientation.Portrait;

converter.Options.MarginLeft = 57;

converter.Options.MarginRight = 28;

converter.Options.MarginTop = 28;

converter.Options.MarginBottom = 28;

var docHtml = converter.ConvertUrl(Server.MapPath($"~/App\_Data/{order.order\_number}.html"));

docHtml.Save(Server.MapPath($"~/App\_Data/orders/{order.order\_number}.pdf"));

docHtml.Close();

System.IO.File.Delete(Server.MapPath($"~/App\_Data/{order.order\_number}.html"));

}

catch (Exception e)

{

Console.Error.WriteLine(e.Message);

throw;

}

return new FilePathResult(Server.MapPath($"~/App\_Data/orders/{order.order\_number}.pdf"),

"application/pdf")

{

FileDownloadName = Path.GetFileName(Server.MapPath($"~/App\_Data/orders/{order.order\_number}.pdf"))

};

}

return null;

}

// GET: Orders

[Authorize]

public ActionResult Index()

{

ViewBag.IsActive = "Orders";

var user = UserManager.FindById(User.Identity.GetUserId());

var orders = \_db.Orders.Include(d => d.Departments).Where(o => o.Departments.name == user.department);

return View(orders.ToList());

}

protected override void Dispose(bool disposing)

{

if (disposing) \_db.Dispose();

base.Dispose(disposing);

}

}

}

@model SparepartsManagementSystem.Models.OrderedSparepartModel

@using (Html.BeginForm("Create", "OrderedSparepart",new { ViewBag.orderid } ,FormMethod.Post,new {@id = "addSpare", @name = "addSpare"}))

{

@Html.AntiForgeryToken()

<h4>OrderedSparepartModel</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Plantid, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("plantid", ViewBag.plantid as SelectList, "Please select plant", new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Plantid, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Sparepartid, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("sparepartid", new SelectList(string.Empty, "sparepartid", "ExtendedInfo"), "Please select sparepart", new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Sparepartid, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Quantity, htmlAttributes: new { @class = "col-form-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })

</div>

</div>

<div class="container">

@Html.HiddenFor(model => model.PlantName)

@Html.HiddenFor(model => model.ExtendedInfo)

</div>

<div class="form-group">

<div class="offset-md-2 col-md-10">

<input type="submit" value="Create" class="btn btn-outline-success" id="btnSubm" name="btnSubm"/>

</div>

</div>

}

<div class="container">

@Html.ActionLink("Back to List", "Index", new { ViewBag.orderid }, new { @class = "btn btn-link" }) |

@Html.ActionLink("Back to list of orders", "Index", new { Controller = "Orders" }, new { @class = "btn btn-link" })

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

<script type="text/javascript">

$(document).ready(function () {

$("#plantid").change(function() {

$("#sparepartid").empty();

$.ajax({

type: 'POST',

url: '@Url.Action("GetSpareparts")',

dataType: 'json',

data: { plantid: $("#plantid").val() },

success: function(spareparts) {

$.each(spareparts,

function(i, item) {

$("#sparepartid").append('<option value="' +

item.Value +

'">' +

item.Text +

'</option>');

});

},

error: function(ex) {

alert('Failed to retrieve spare parts.' + ex);

}

});

return false;

});

$("#Quantity").change(function () {

$("#PlantName").val($("#plantid :selected").text());

$("#ExtendedInfo").val($("#sparepartid :selected").text());

});

});

</script>

}

## Приложение №7

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using System.Security.Claims;

using System.Threading.Tasks;

namespace SparepartsManagementSystem.Models

{

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection", throwIfV1Schema: false)

{

}

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

}

// В профиль пользователя можно добавить дополнительные данные, если указать больше свойств для

// класса ApplicationUser. Подробности см. на странице https://go.microsoft.com/fwlink/?LinkID=317594.

public class ApplicationUser : IdentityUser

{

public string department { get; set; }

public string firstName { get; set; }

public string lastName { get; set; }

public string position { get; set; }

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Обратите внимание, что authenticationType должен совпадать с типом, определенным в CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Здесь добавьте утверждения пользователя

userIdentity.AddClaim(new Claim("Department", department));

userIdentity.AddClaim(new Claim("Occupation", position));

userIdentity.AddClaim(new Claim("FirstName", firstName));

userIdentity.AddClaim(new Claim("LastName", lastName));

return userIdentity;

}

}

}

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using SparepartsManagementSystem.Models;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

namespace SparepartsManagementSystem.Controllers

{

[Authorize]

public class AccountController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public AccountController()

{

}

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get => \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

private set => \_signInManager = value;

}

public ApplicationUserManager UserManager

{

get => \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

private set => \_userManager = value;

}

// GET: /Account/ConfirmEmail

[AllowAnonymous]

public async Task<ActionResult> ConfirmEmail(string userId, string code)

{

if (userId == null || code == null) return View("Error");

var result = await UserManager.ConfirmEmailAsync(userId, code);

return View(result.Succeeded ? "ConfirmEmail" : "Error");

}

// POST: /Account/ExternalLogin

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult ExternalLogin(string provider, string returnUrl)

{

// Запрос перенаправления к внешнему поставщику входа

return new ChallengeResult(provider,

Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl }));

}

// GET: /Account/ExternalLoginCallback

[AllowAnonymous]

public async Task<ActionResult> ExternalLoginCallback(string returnUrl)

{

var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();

if (loginInfo == null) return RedirectToAction("Login");

// Выполнение входа пользователя посредством данного внешнего поставщика входа, если у

// пользователя уже есть имя входа

var result = await SignInManager.ExternalSignInAsync(loginInfo, false);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(returnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.RequiresVerification:

return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = false });

case SignInStatus.Failure:

default:

// Если у пользователя нет учетной записи, то ему предлагается создать ее

ViewBag.ReturnUrl = returnUrl;

ViewBag.LoginProvider = loginInfo.Login.LoginProvider;

return View("ExternalLoginConfirmation",

new ExternalLoginConfirmationViewModel { Email = loginInfo.Email });

}

}

// POST: /Account/ExternalLoginConfirmation

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model,

string returnUrl)

{

if (User.Identity.IsAuthenticated) return RedirectToAction("Index", "Manage");

if (ModelState.IsValid)

{

// Получение сведений о пользователе от внешнего поставщика входа

var info = await AuthenticationManager.GetExternalLoginInfoAsync();

if (info == null) return View("ExternalLoginFailure");

var user = new ApplicationUser { UserName = model.Email, Email = model.Email };

var result = await UserManager.CreateAsync(user);

if (result.Succeeded)

{

result = await UserManager.AddLoginAsync(user.Id, info.Login);

if (result.Succeeded)

{

await SignInManager.SignInAsync(user, false, false);

return RedirectToLocal(returnUrl);

}

}

AddErrors(result);

}

ViewBag.ReturnUrl = returnUrl;

return View(model);

}

// GET: /Account/ExternalLoginFailure

[AllowAnonymous]

public ActionResult ExternalLoginFailure()

{

return View();

}

// GET: /Account/ForgotPassword

[AllowAnonymous]

public ActionResult ForgotPassword()

{

return View();

}

// POST: /Account/ForgotPassword

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)

{

if (ModelState.IsValid)

{

var user = await UserManager.FindByNameAsync(model.Email);

if (user == null || !await UserManager.IsEmailConfirmedAsync(user.Id))

return View("ForgotPasswordConfirmation");

// Дополнительные сведения о включении подтверждения учетной записи и сброса пароля

// см. на странице https://go.microsoft.com/fwlink/?LinkID=320771. Отправка сообщения

// электронной почты с этой ссылкой string code = await

// UserManager.GeneratePasswordResetTokenAsync(user.Id); var callbackUrl =

// Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code },

// protocol: Request.Url.Scheme); await UserManager.SendEmailAsync(user.Id, "Сброс

// пароля", "Сбросьте ваш пароль, щелкнув <a href=\"" + callbackUrl +

// "\">здесь</a>"); return RedirectToAction("ForgotPasswordConfirmation", "Account");

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

return View(model);

}

// GET: /Account/ForgotPasswordConfirmation

[AllowAnonymous]

public ActionResult ForgotPasswordConfirmation()

{

return View();

}

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

ViewBag.IsActive = "login";

return View();

}

// POST: /Account/Login

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)

{

if (!ModelState.IsValid)

{

ViewBag.IsActive = "login";

return View(model);

}

// Сбои при входе не приводят к блокированию учетной записи Чтобы ошибки при вводе пароля

// инициировали блокирование учетной записи, замените на shouldLockout: true

var result =

await SignInManager.PasswordSignInAsync(model.Occupation, model.Password, model.RememberMe, false);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(returnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.RequiresVerification:

return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, model.RememberMe });

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Failed login attempt.");

return View(model);

}

}

// POST: /Account/LogOff

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LogOff()

{

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);

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

}

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

ViewBag.IsActive = "register";

return View();

}

// POST: /Account/Register

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new ApplicationUser

{

UserName = model.Occupation,

Email = model.Email,

firstName = model.Name,

lastName = model.Surname,

department = model.Department,

position = model.Occupation

};

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

if (result.Succeeded)

{

await SignInManager.SignInAsync(user, false, false);

// Дополнительные сведения о включении подтверждения учетной записи и сброса

// пароля см. на странице https://go.microsoft.com/fwlink/?LinkID=320771.

// Отправка сообщения электронной почты с этой ссылкой string code = await

// UserManager.GenerateEmailConfirmationTokenAsync(user.Id); var callbackUrl =

// Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code },

// protocol: Request.Url.Scheme); await UserManager.SendEmailAsync(user.Id,

// "Подтверждение учетной записи", "Подтвердите вашу учетную запись, щелкнув <a

// href=\"" + callbackUrl + "\">здесь</a>");

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

}

AddErrors(result);

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

ViewBag.IsActive = "register";

return View(model);

}

// GET: /Account/ResetPassword

[AllowAnonymous]

public ActionResult ResetPassword(string code)

{

return code == null ? View("Error") : View();

}

// POST: /Account/ResetPassword

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)

{

if (!ModelState.IsValid) return View(model);

var user = await UserManager.FindByNameAsync(model.Email);

if (user == null) return RedirectToAction("ResetPasswordConfirmation", "Account");

var result = await UserManager.ResetPasswordAsync(user.Id, model.Code, model.Password);

if (result.Succeeded) return RedirectToAction("ResetPasswordConfirmation", "Account");

AddErrors(result);

return View();

}

// GET: /Account/ResetPasswordConfirmation

[AllowAnonymous]

public ActionResult ResetPasswordConfirmation()

{

return View();

}

// GET: /Account/SendCode

[AllowAnonymous]

public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)

{

var userId = await SignInManager.GetVerifiedUserIdAsync();

if (userId == null) return View("Error");

var userFactors = await UserManager.GetValidTwoFactorProvidersAsync(userId);

var factorOptions = userFactors.Select(purpose => new SelectListItem { Text = purpose, Value = purpose })

.ToList();

return View(new SendCodeViewModel

{

Providers = factorOptions,

ReturnUrl = returnUrl,

RememberMe = rememberMe

});

}

// POST: /Account/SendCode

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> SendCode(SendCodeViewModel model)

{

if (!ModelState.IsValid) return View();

// Создание и отправка маркера

if (!await SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider)) return View("Error");

return RedirectToAction("VerifyCode",

new { Provider = model.SelectedProvider, model.ReturnUrl, model.RememberMe });

}

// GET: /Account/VerifyCode

[AllowAnonymous]

public async Task<ActionResult> VerifyCode(string provider, string returnUrl, bool rememberMe)

{

// Требовать предварительный вход пользователя с помощью имени пользователя и пароля или

// внешнего имени входа

if (!await SignInManager.HasBeenVerifiedAsync()) return View("Error");

return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl, RememberMe = rememberMe });

}

// POST: /Account/VerifyCode

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)

{

if (!ModelState.IsValid) return View(model);

// Приведенный ниже код защищает от атак методом подбора, направленных на двухфакторные

// коды. Если пользователь введет неправильные коды за указанное время, его учетная

// запись будет заблокирована на заданный период. Параметры блокирования учетных записей

// можно настроить в IdentityConfig

var result = await SignInManager.TwoFactorSignInAsync(model.Provider, model.Code, model.RememberMe,

model.RememberBrowser);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(model.ReturnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Incorrect code.");

return View(model);

}

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

if (\_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

if (\_signInManager != null)

{

\_signInManager.Dispose();

\_signInManager = null;

}

}

base.Dispose(disposing);

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager => HttpContext.GetOwinContext().Authentication;

private void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors) ModelState.AddModelError("", error);

}

private ActionResult RedirectToLocal(string returnUrl)

{

if (Url.IsLocalUrl(returnUrl)) return Redirect(returnUrl);

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

}

internal class ChallengeResult : HttpUnauthorizedResult

{

public ChallengeResult(string provider, string redirectUri)

: this(provider, redirectUri, null)

{

}

public ChallengeResult(string provider, string redirectUri, string userId)

{

LoginProvider = provider;

RedirectUri = redirectUri;

UserId = userId;

}

public string LoginProvider { get; set; }

public string RedirectUri { get; set; }

public string UserId { get; set; }

public override void ExecuteResult(ControllerContext context)

{

var properties = new AuthenticationProperties { RedirectUri = RedirectUri };

if (UserId != null) properties.Dictionary[XsrfKey] = UserId;

context.HttpContext.GetOwinContext().Authentication.Challenge(properties, LoginProvider);

}

}

#endregion Вспомогательные приложения

}

}

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using SparepartsManagementSystem.Models;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

namespace SparepartsManagementSystem.Controllers

{

[Authorize]

public class ManageController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public ManageController()

{

}

public ManageController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get => \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

private set => \_signInManager = value;

}

public ApplicationUserManager UserManager

{

get => \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

private set => \_userManager = value;

}

// GET: /Manage/AddPhoneNumber

public ActionResult AddPhoneNumber()

{

return View();

}

// POST: /Manage/AddPhoneNumber

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)

{

if (!ModelState.IsValid) return View(model);

// Создание и отправка маркера

var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), model.Number);

if (UserManager.SmsService != null)

{

var message = new IdentityMessage

{

Destination = model.Number,

Body = "Ваш код безопасности: " + code

};

await UserManager.SmsService.SendAsync(message);

}

return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber = model.Number });

}

// GET: /Manage/ChangePassword

public ActionResult ChangePassword()

{

return View();

}

// POST: /Manage/ChangePassword

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (!ModelState.IsValid) return View(model);

var result =

await UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword, model.NewPassword);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

return RedirectToAction("Index", new { Message = ManageMessageId.ChangePasswordSuccess });

}

AddErrors(result);

return View(model);

}

// POST: /Manage/DisableTwoFactorAuthentication

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DisableTwoFactorAuthentication()

{

await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

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

}

// POST: /Manage/EnableTwoFactorAuthentication

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> EnableTwoFactorAuthentication()

{

await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

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

}

// GET: /Manage/Index

public async Task<ActionResult> Index(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.ChangePasswordSuccess

? "Ваш пароль изменен."

: message == ManageMessageId.SetPasswordSuccess

? "Пароль задан."

: message == ManageMessageId.SetTwoFactorSuccess

? "Настроен поставщик двухфакторной проверки подлинности."

: message == ManageMessageId.Error

? "Произошла ошибка."

: message == ManageMessageId.AddPhoneSuccess

? "Ваш номер телефона добавлен."

: message == ManageMessageId.RemovePhoneSuccess

? "Ваш номер телефона удален."

: "";

var userId = User.Identity.GetUserId();

var model = new IndexViewModel

{

HasPassword = HasPassword(),

PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),

TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),

Logins = await UserManager.GetLoginsAsync(userId),

BrowserRemembered = await AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)

};

return View(model);

}

// POST: /Manage/LinkLogin

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LinkLogin(string provider)

{

// Запрос перенаправления к внешнему поставщику входа для связывания имени входа текущего пользователя

return new AccountController.ChallengeResult(provider, Url.Action("LinkLoginCallback", "Manage"),

User.Identity.GetUserId());

}

// GET: /Manage/LinkLoginCallback

public async Task<ActionResult> LinkLoginCallback()

{

var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());

if (loginInfo == null) return RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });

var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);

return result.Succeeded

? RedirectToAction("ManageLogins")

: RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });

}

// GET: /Manage/ManageLogins

public async Task<ActionResult> ManageLogins(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.RemoveLoginSuccess ? "Внешнее имя входа удалено."

: message == ManageMessageId.Error ? "Произошла ошибка."

: "";

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user == null) return View("Error");

var userLogins = await UserManager.GetLoginsAsync(User.Identity.GetUserId());

var otherLogins = AuthenticationManager.GetExternalAuthenticationTypes()

.Where(auth => userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();

ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;

return View(new ManageLoginsViewModel

{

CurrentLogins = userLogins,

OtherLogins = otherLogins

});

}

// POST: /Manage/RemoveLogin

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)

{

ManageMessageId? message;

var result = await UserManager.RemoveLoginAsync(User.Identity.GetUserId(),

new UserLoginInfo(loginProvider, providerKey));

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

message = ManageMessageId.RemoveLoginSuccess;

}

else

{

message = ManageMessageId.Error;

}

return RedirectToAction("ManageLogins", new { Message = message });

}

// POST: /Manage/RemovePhoneNumber

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> RemovePhoneNumber()

{

var result = await UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);

if (!result.Succeeded) return RedirectToAction("Index", new { Message = ManageMessageId.Error });

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

return RedirectToAction("Index", new { Message = ManageMessageId.RemovePhoneSuccess });

}

// GET: /Manage/SetPassword

public ActionResult SetPassword()

{

return View();

}

// POST: /Manage/SetPassword

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> SetPassword(SetPasswordViewModel model)

{

if (ModelState.IsValid)

{

var result = await UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

return RedirectToAction("Index", new { Message = ManageMessageId.SetPasswordSuccess });

}

AddErrors(result);

}

// Это сообщение означает наличие ошибки; повторное отображение формы

return View(model);

}

// GET: /Manage/VerifyPhoneNumber

public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)

{

var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), phoneNumber);

// Отправка SMS через поставщик SMS для проверки номера телефона

return phoneNumber == null

? View("Error")

: View(new VerifyPhoneNumberViewModel { PhoneNumber = phoneNumber });

}

// POST: /Manage/VerifyPhoneNumber

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel model)

{

if (!ModelState.IsValid) return View(model);

var result =

await UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(), model.PhoneNumber, model.Code);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null) await SignInManager.SignInAsync(user, false, false);

return RedirectToAction("Index", new { Message = ManageMessageId.AddPhoneSuccess });

}

// Это сообщение означает наличие ошибки; повторное отображение формы

ModelState.AddModelError("", "Не удалось проверить телефон");

return View(model);

}

protected override void Dispose(bool disposing)

{

if (disposing && \_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

base.Dispose(disposing);

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

private const string XsrfKey = "XsrfId";

public enum ManageMessageId

{

AddPhoneSuccess,

ChangePasswordSuccess,

SetTwoFactorSuccess,

SetPasswordSuccess,

RemoveLoginSuccess,

RemovePhoneSuccess,

Error

}

private IAuthenticationManager AuthenticationManager => HttpContext.GetOwinContext().Authentication;

private void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors) ModelState.AddModelError("", error);

}

private bool HasPassword()

{

var user = UserManager.FindById(User.Identity.GetUserId());

if (user != null) return user.PasswordHash != null;

return false;

}

private bool HasPhoneNumber()

{

var user = UserManager.FindById(User.Identity.GetUserId());

if (user != null) return user.PhoneNumber != null;

return false;

}

#endregion Вспомогательные приложения

}

}

# Лист замечаний

|  |  |
| --- | --- |
| Ф.И.О. | Суть замечаний, оценка и подпись |
|  |  |