**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

**Отчет по производственной (технологической**) **практике**

на тему: «Web-приложение «Автоматизация СТО» на базе технологии ASP»

Исполнитель: студент гр. ИП-31

Процкая М. А.

Руководитель от предприятия:

начальник отдела

Журова И. И.

Руководитель: ст. преподаватель Богданова Н. С.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание**

Введение………………………………………………………………………….......3

1 Общие сведения об организации………………………………………………….5

* 1. История организации…………………………………………………..5
  2. Охрана труда и техника безопасности на рабочем месте…………....8
  3. Технологии, используемые на предприятии, программное обеспечение на предприятии………………………………………………………10

1. Индивидуальное задание………………………………………………………12
   1. Постановка задачи…………………………………………………….12
   2. Моделирования базы данных………………………………………...12
   3. Логическая и физическая модели базы данных……………………..15
   4. Описание технологии решения поставленной задачи………………16
   5. Описание алгоритма решения поставленной задачи………………..18
   6. Структура приложения……………………………………………….19
   7. Общие сведения для работы с приложением………………………..21
   8. Описание интерфейса пользователя…………………………………21
   9. Тестирование приложения……………………………………………26

Заключение………………………………………………………………………….28

Список использованной литературы………………………………………………29

Приложение А………………………………………………………………………30

# Введение

Задача накопления, обработки и распространения информации стояла перед человечеством на всех этапах его развития. В течение долгого времени основными инструментами для ее решения были мозг, язык и слух человека. Первое кардинальное изменение произошло с приходом письменности, а затем изобретением книгопечатания.

Хранение информации в памяти электронной вычислительной машины (ЭВМ) придает этой информации принципиально новое качество динамичности, т.е. способности к быстрой перестройке и непосредственному ее использованию в решаемых на ЭВМ задачах. Устройства автоматической печати, которыми снабжены современные ЭВМ, позволяют в случае необходимости быстро представить любую выборку из этой информации в форме представления на бумаге.

По мере своего дальнейшего развития административные системы обработки данных переросли в автоматизированные системы управления соответствующими объектами, в которых, как правило, не ограничиваются одной ЭВМ, а в составе двух и более ЭВМ объединяют в вычислительный комплекс.

Интеллектуальные автоматизированные рабочие места (АРМ) в свою очередь также содержат в своем составе ЭВМ, тем или иным способом подсоединенную к центральной автоматизированной системе управления (АСУ). Устройства ввода также должны обеспечивать широкий спектр вводимой информации: текстовой, координатной, факсимильной и т.д. Поэтому АРМ оснащаются при необходимости универсальной или специальной клавиатурой, устройствами ввода координатной информации, различного рода сканерами и т.д.

Развитие рынка автоиндустрии непрерывно развивается и постоянно находится в состоянии трансформации. Трансформация происходит в зависимости от потребностей потребителей данной услуги. Постоянно расширяется сфера услуг и уровень обслуживания.

С каждым годов в Беларуси увеличивается количество автовладельцев, которые предпочитают проходить качественное техническое обслуживание своего автомобиля.

Рост общего количества личных легковых автомобилей по стране за период с 2000 года выглядит следующим образом: их количество увеличилось с 1 385,9 тыс. единиц в 2000-м до 1 737,1 тыс. в 2005-м. В 2010-м насчитывалось уже 2 501,2 тыс., в 2015-м – 2 920,2 тыс., в 2017-м – 2 972,7 тыс. и в конце 2018-го – 3 031 тыс. То есть, с 2000-го количество личных авто выросло более чем вдвое [5].

Согласно правилам проведения государственного технического осмотра механических транспортных средств и прицепов к ним [6], технический осмотр транспортных средств проводится на диагностических станциях Госавтоинспекции (ГАИ), пунктах технического контроля, а также непосредственно на предприятиях. Для проведения технического осмотра транспортных средств на предприятиях, при получении разрешения ГАИ, создаются комиссии.

Поэтому спрос на услуги станций технического обслуживания растет, что говорит о несомненной актуальности тематики курсовой работы.

Станция технического обслуживания (СТО) — организация, предоставляющая услуги населению по плановому техническому обслуживанию, текущему и капитальному ремонтам, устранению неисправностей, установке дополнительного оборудования, восстановительному ремонту автотранспорта.

С точки зрения функционирования или структуры СТО, можно сказать, что она оказывает услуги схожего типа, в частности производит какие-либо манипуляции с автомашиной, приводя ее в рабочее состояние, либо совершенствуя какие-либо ее части.

Все обращающиеся в СТО клиенты, желающие оформить заказ на оказание услуг, должны заполнить форму регистрации, в которой необходимо указать свои ФИО, марку автомобиля, номерной знак и ФИО владельца машины, контактный телефон.

Таким образом, в функционирование СТО входит:

* регистрация клиентов,
* прием заказа на оказание услуг,
* справочная информация о сделанном заказе,
* отчет о проделанных работах и расчет стоимости предоставленных услуг.

Данные функции на СТО выполняет администратор. Это очень трудоемкая бумажная работа, затрачивающая большое количество времени, поэтому необходимо было разработать АРМ администратора.

Следуя из вышесказанного, можно сказать, что задачами данной работы являются значительное ускорение и повышение качества работы СТО, уменьшение трудоемкости оформления заказа, решение проблем с потерей данных о клиентах, автомобилях, деталях и т.п.

Цель работы – автоматизация рабочего места администратора СТО, которым смог бы воспользоваться любой пользователь, имеющий базовые умения для работы с компьютером.

1. **ОБЩИЕ СВЕДЕНИЯ О ПРЕДПРИЯТИИ**

*IBA Group* — альянс компаний, специализирующихся в области информационных технологий, разработчиков, производителей и поставщиков решений и услуг в ИТ-сфере в странах Центральной и Восточной Европы.

Под брендом *IBA Group* работает более 20 компаний, расположенных в Чехии, Беларуси, России, Германии, США, Казахстане, Великобритании, Украине, Словакии, ЮАР, Болгарии, Израиле и в Саудовской Аравии. В 2005 году после вступления Чехии в Европейский союз местом размещения головного офиса *IBA Group* стала Прага.

Деятельность *IBA Group* включает реализацию проектов в области системной интеграции, разработку программного обеспечения (ПО), ИТ-консалтинг и обучение, производство и послегарантийное обслуживание средств вычислительной техники (СВТ).

*IBA Group* выполнила более 2000 проектов для заказчиков, сфера деятельности которых относится к самым различным секторам экономики, включая информационные технологии, транспорт, телекоммуникации, энергетику.

* 1. **История организации**

1993 г. – Создание компании *IBA* (СП ЗАО «Международный деловой альянс», г. Минск, Беларусь).

1994 г. – Открытие Вычислительного центра *IBA* (г. Минск, Беларусь).

1997 г. – Создание СП «Информационные производственные архитектуры» – *IBA IS* (г. Минск, Беларусь).

1998 г. – Открытие офиса *IBA USA* (*Mountain View*, США).

1999 г. – Открытие центра разработки *IBA CZ* (г. Прага, Чехия); офиса *IBA* *Development* (г. Лимассол, Кипр); Технического Центра *IBA* (г. Минск, Беларусь).

2000 г. – Открытие офиса *IBA* *IT GmbH* (г. Дюссельдорф, Германия); открытие студии *Web*-дизайна *PixelHead* (г. Минск, Беларусь).

2001 г. – Открытие центра разработки *IBA Gomel* (г. Гомель, Беларусь).

2002 г. – Открытие Учебного центра *IBA* (г. Минск, Беларусь); вступление *IBA* в Ассоциацию Разработчиков Программного Обеспечения РУССОФТ.

2003 г. – Открытие отделения *IBA* в г. Новополоцк, Беларусь.

2004 г. – Открытие отделения *IBA* в г. Могилев, Беларусь.

2005 г. – Открытие головного офиса *IBA Group* (г. Прага, Чехия), открытие отделения *IBA CZ* (г. Брно, Чехия); вступление *IBA Group* в Научно-технологическую ассоциацию «Инфопарк».

2006 г. – Открытие офиса *IBA Rus* (г. Москва, Россия); создание компании «*IT* парк» (ныне *IBA IT Park*) (г. Минск, Беларусь).

2007 г. – *IBA IT Park* – резидент ПВТ.

2008 – Открытие нового офиса *IBA CZ* (г. Брно, Чехия); создание *IBA Gomel Park* (г. Гомель, Беларусь).

2009 г. – *IBA Gomel Park* — резидент ПВТ.

2010 г. – Сертификат соответствия ОАЦ на собственный программный продукт «Канцлер» Офис *IBA Group* начал свою работу в Астане, Казахстан.

2011 г. – Центр разработок *IBA Group* — ТОО «*IBA* Казахстан» — открыт в столице Казахстана г. Астане Гомельский центр разработок *IBA Group* отметил свой десятилетний юбилей. Открытие офиса *IBA UK* в г. Лондоне, Великобритания.

2012 г. – Открытие нового предприятия *IBA Ukraine* в г. Киеве, Украина. Торжественная церемония закладки капсулы с памятной грамотой в фундамент строящегося на территории ПВТ производственно-жилого кластера *IBA Group*.

2013 г. – Компании *IBA Minsk* вручен сертификат соответствия Системы менеджмента информационной безопасности требованиям стандарта СТБ *ISO/IEC* 27001-2011. В марте 2013 года *IBA Minsk* награждён Почетной грамотой Совета Министров Республики Беларусь за большие достижения в развитии и внедрении информационных технологий РБ Предприятие *IBA IT Park*, один из белорусских центров разработок *IBA Group*, — в списке крупнейших компаний-резидентов ПВТ по итогам первого полугодия 2013 года.

2014 г. – *IBA Group* вывела на рынок новый программный продукт «*Start Point*» на базе платформ PTC *Creo* и PTC *Windchill* для создания автоматизированной системы разработки изделия. *IBA Minsk* — официальный дистрибутор компании *SafeNet* на территории Республики Беларусь. *IBA IT Park*, один из центров разработок *IBA Group*, удостоен Национальной премии «Золотой байт-2014» в номинации «Крупнейший поставщик решений для белорусского рынка».

2015 г. – *IBA Group* включена в категорию «Лидеры» рейтинга «*The* 2015 *Global Outsourcing* 100». *IBA IT Park*, один из белорусских центров разработок *IBA Group*, – среди крупнейших компаний-резидентов ПВТ по итогам 2014 года Система автоматизации оплаты в общественном транспорте Минска, созданная *IBA Group*, признана лучшим отраслевым решением года на конкурсе «*European IT & Software Excellence Awards*» *IBA Group* открыла новый офис в г. Братислава (Словакия) *IBA CZ* открыла новый центр разработок в Остраве, третьем по величине городе Чешской Республики Компания *IBA Gomel Park* открыла новое производственное здание Система оплаты и контроля проезда в общественном транспорте, разработанная *IBA Group*, прошла в финал конкурса Европейской ассоциации аутсорсинга Начал свою работу Центр обработки данных *IBA Group* (ЦОД), находящийся на территории кластера *IBA Group* в Парке высоких технологий в г. Минске 55 *IBA Group* одержала победу в номинации «Выбор профессионального сообщества», получила второе место в номинации «Республика» в рамках «Премии HR-бренд Беларусь 2015» .

2016 г. – *IBA Group* – в категории «Лидеры» рейтинга «*The* 2016 *Global Outsourcing* 100», а также отмечена как лучшая компания (*Top Company*) за достижения в социальной ответственности, высокую удовлетворенность клиентов, инновации, сертификацию и победы в конкурсах и рейтингах. *IBA Group* включена в список финалистов европейского конкурса ИТ-проектов «*European IT & Software Excellence Awards* 2016». *IBA Minsk* (СП ЗАО «Международный деловой альянс»), центр разработок *IBA Group*, — пятикратный лауреат Премии Правительства Республики Беларусь за достижения в области качества. *IBA IT Park* подтвердила звание Лауреата. Впервые в Белоруссии в Беларусбанке запущен в эксплуатацию валютно-обменный терминал производства *IBA Group*. 8 июня *IBA Group* открыла научно-производственный кластер на территории Парка высоких технологий. *IBA Group* отмечена в четырех категориях рейтинга «*The* 2016 *Global Outsourcing* 100». Победа в конкурсе Глобальной сорсинговой ассоциации «2016 *EOA Awards*» и включение в рейтинг «*Software* 500» журнала «*Software Magazine*». *IBA Group* и Беларусбанк запустили мобильный эквайринг с поддержкой всех видов банковских карт, включая бесконтактные. Совместный проект *IBA Group* и ОАО «Газпром трансгаз Беларусь» по созданию и внедрению Системы электронного документооборота в части управления документацией ТОРО — Бронзовый призер конкурса *SAP Quality Awards* 2016 в регионе СНГ в категории «Быстрый результат».

2017 г. – *IBA Group* – победитель конкурса *CEE Shared Services & Outsourcing Awards.* Проект *IBA Group* стал победителем конкурса «*European IT & Software Excellence Awards* 2017». Проект *IBA Institute* – победитель конкурса «*BestinCSR*». *IBA Institute* отметил свое 15-летие. *IBA Group* – «суперзвезда» рейтинга «*Global Outsourcing* 100» и лучшая компания в четырех категориях. *IBA Group* празднует новоселье в пражском офисе. АСОКП продолжает развиваться: в минском метрополитене стала доступной оплата проезда бесконтактными банковскими картами. *IBA Group* подписала Договор о КСО в Белоруссии. *IBA Group* включена в рейтинг «*Software* 500» 2017 года.

2018 г. – *IBA Group* – в рейтинге *The* 2018 *Global Outsourcing* 100. *IBA Group* – в финале конкурса «*European IT & Software Excellence Awards* 2018». Открытие нового центра разработок *IBA Group* в Болгарии. *IBA Group* – победитель конкурса «*European IT & Software Excellence Award* 2018». *IBA Group* отметила свой 25-летний юбилей. С 12 июня 2018 года на всех станциях минского метро заработала оплата проезда бесконтактной банковской картой. *IBA Group* – «суперзвезда» и лучшая компания в четырех категориях рейтинга «*The* 2018 *Global Outsourcing* 100». Победитель конкурса «*GSA Awards* 2018 Сразу два решения *IBA Group* – в финале конкурса *SAP Coder* 2018. *IBA Group* получила 315-е место в рейтинге крупнейших компаний мира «*Software* 500» 2018 года.

2019 г. – *IBA Group* вышла в финал «*IT Excellence Awards* 2019». Компания в рейтинге «*The Best of The Global Outsourcing* 100». *IBA Group* стала шестикратным лауреатом Премии Правительства Республики Беларусь за достижения в области качества. *IBA Group* включена в список *Best of The Global Outsourcing* 100.

* 1. Охрана труда и техника безопасности на рабочем месте

Охрана труда – система обеспечения безопасности жизни и здоровья работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

Руководством ООО «ИВА-Гомель-Парк» уделяется большое внимание улучшению эргономики рабочих мест, обеспечению гигиены и совершенствованию организации труда, регламентации режимов труда и отдыха.

Вся деятельность в области охраны труда на ООО «ИВА-Гомель-Парк» регламентирована действующим законодательством Республики Беларусь, санитарными нормами и правилами, гигиеническими нормативами, предписаниями надзорных органов.

Важным фактором создания безопасных условий труда является оптимизация организации рабочих мест. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и целесообразно организованное в отношении пространства, формы, размера, обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда инженера возрастает с 8 до 20 процентов.

Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования».

Требования к организации работы при использовании персонального компьютера и организационных средств определяются СанПиН 9-131 РБ 2000 «Гигиенические требования к видеодисплейным терминалам, электронно-вычисли-тельным машинам и организации работы».

Согласно ГОСТ 12.2.032-78, конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

* оптимальное размещение оборудования, входящего и состав рабочего места;
* достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
* уровень акустического шума не должен превышать допустимого значения.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Помещения для работы программиста должны иметь естественное и искусственное освещение.

Искусственное освещение в помещениях эксплуатации ВДТ и ПЭВМ должно осуществляться системой общего равномерного освещения. В административно-общественных помещениях, в случаях преимущественной работы с документами, допускается применение системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в тоне размещения рабочего документа должна быть 300-500 лк. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей (размер ВДТ и ПЭВМ, клавиатуры и др.), характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Тип рабочего стула (кресла) должен выбираться в зависимости от характера и продолжительности работы.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм, при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии не менее чем 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Помимо требований к организации рабочего места СанПиН 9-131 РБ 2000 устанавливает требования к микроклимату рабочей зоны: влажности, температуре, скорости потока воздуха и пр.

Техника безопасности — это система организационных мероприятий и технических средств, предотвращающих воздействие на работающих опасных производственных факторов.

1.3 Технологии, используемые на предприятии, программное обеспечение

а) Проектирование, разработка и сопровождение:

1. приложений для платформы *Java Enterprise Edition, Microsoft SharePoint/.Net*;
2. приложений для извлечения, преобразования, загрузки массивов данных с использованием *ETL*-технологий;
3. приложений в среде «облачных» вычислений;
4. приложений для работы с *BigData* и *SemanticWeb*;
5. приложений для мобильных устройств на различных платформах (*Android, iOS,Windows Phone/Mobile*);
6. приложений с использованием концепции *SPA (Single Page Application)* и сервис-ориентированной архитектуры (*SOA*);
7. приложений на языке *ABAP* для систем *SAP*;
8. приложений на базе системы коллективной работы *IBM Notes/Domino*;
9. продуктов Интернет и *e-business* (информационных порталов, *Web*-сайтов, систем ведения электронных платежей и др.);
10. решений для обмена данными между корпоративными системами, включая унаследованные, *SAP* и *Web*-приложения (на базе *IBM Websphere* *Message Broker*)

б) Создание и развитие интегрированных систем управления предприятием на базе решений *SAP*

в) Миграция программных продуктов между разнообразными платформами и средами

г) Проектирование и администрирование баз данных *(DB2, Oracle, MS SQL Server)*

д) Разработка и внедрение *BI*-решений по отчетности и анализу данных

е) Разработка облачных сервисов по управлению *НСИ (MDM, MDG),* в том числе:

1. анализ качества данных в справочниках (*ISO* 8000/22745);
2. построение систем управления НСИ (*SAP* *MDG*, *MDM/BPM/Webdynpro*);
3. построение онтологий и средств структуризации данных (RDF/OWL);
4. преобразование данных НСИ к формату *Semantic Web* и организация доступа (*SPARQL*)

ж) Банковские технологии.

1. **ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**
   1. **Постановка задачи**

Необходимо реализовать *Web*-приложение на тему «Автоматизация СТО».

Есть 2 типа пользователей: администратор и работник:

* администратор ведет учет всей справочной информации, может получить информацию о выполненных заказах;
* работник оформляет заказы и отмечает выполнение.

Должны быть реализованы регистрация и авторизация.

Основные страницы:

* + регистрация/авторизация;
  + добавление/удаление/редактирование информации о владельцах, автомобилях и сотрудниках СТО;
  + оформление заказов;
  + отчет о выполненных заказах.
  1. **Моделирование базы данных**

Моделирование данных – важный этап построения хорошо функционирующей базы данных. Чтобы база данных справлялась с обеспечением бизнеса необходимой информацией, ей требуется хороший проект и надежное основание, то есть модель данных.

База данных – это совместно используемый набор логически связанных и структурированных данных, относящихся к одной предметной области.

Предметная область – часть реального мира, подлежащая изучению с целью организации управления.

Предметная область представляется множеством фрагментов, например, книжный магазин может быть представлен в виде совокупности подразделений ­­– торговый зал, склад, аппарат управления, бухгалтерия и т.д. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, характеризуемых различными взглядами на предметную область. Такой взгляд, называемый представлением, содержит ключевые и наиболее важные с точки зрения этого пользователя аспекты предметной области, а несущественные аспекты из рассмотрения исключены.

База данных представляет собой хранилище, содержащее не только информацию, но и описание структуры данных. Такое описание обычно называют словарем данных, а сами элементы описания – метаданными, то есть данными о данных. Словарь данных обеспечивает основной принцип организации базы данных – независимость данных от используемого программного обеспечения.

С понятием базы данных неразрывно связано понятие системы управления базами данных. СУБД обычно сочетает в себе автоматизированные средства структуризации и манипулирования данными со средствами обеспечения секретности, восстановления и сохранности информации в многопользовательском окружении.

По способу установления связей между данными и способу манипуляции ими различают три основных модели данных: реляционную, иерархическую и сетевую. Иерархическая и сетевая модели являются устаревшими и используются редко, поэтому изложение всего дальнейшего материала будет проводиться применительно к реляционной модели. Определение реляционной модели будет дано в следующем разделе.

Моделирование данных обычно проводится в три этапа: концептуальный, логический и физический. В таблице 2 представлены основные этапы процесса проектирования базы данных. Каждый из этапов характеризуется определенной последовательностью работ, определенным уровнем представления данных.

Моделирование данных – важнейший этап построения базы данных. Модель данных представляет деловую информацию. Если в модели данных будут дефекты, то дефекты появятся и в самой базе данных, и во всех программах, которые к ней обращаются. Модель данных и саму базу данных следует проектировать таким образом, чтобы обеспечить ее гибкость и перспективы расширения.

Этапы моделирование базы данных описаны в таблице 2.1.

Таблица 2.1 – **Этапы моделирования базы данных**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап | Основные  задачи | Основные шаги | Базовые понятия | Информационное представление |
| 1 | 2 | 3 | 4 | 5 |
| Концептуальный | Сбор, анализ  и  редактирование требований  к данным | Обследование предметной области.  Выявление всех фрагментов предметной области.  По окончании данного этапа получаем концептуальную модель – ER-диаграмму. | Сущности, связи, атрибуты | Аналитика |
| Логический | Преобразование требований  к данным  в структуры | ER-диаграмма преобразуется в набор таблиц, при этом проводится их нормализация. | Таблицы, записи, элементы данных, | Программиста |

Продолжение таблицы 2.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
|  | данных | В результате получаем СУБД ориентированную структуру базы данных и спецификации прикладных программ.  На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей. | связи между записями |  |
| Физический | Определение особенностей хранения  данных,  методов  доступа к  данным и пр. | Адаптация логической модели к выбранной программной платформе.  Выбор и построение индексов.  Организация протоколирования, доступа к данным и др. | Группировка данных, индексы, методы доступа | Администратора |

Первым и важнейшим этапом проектирования базы данных является этап концептуального моделирования предметной области. Его задача - определение понятий о предметах, фактах и событиях, информацию о которых будет хранить, и обрабатывать база данных. Для этого необходимо выделить понятия, с которыми работает пользователь, и заменить их информационными представлениями.

Концептуальная схема, полученная по окончании этапа концептуального моделирования, является полной совокупностью всех требований к данным, полученным на основе пользовательских представлений о реальном мире.

* 1. **Логическая и физическая модели базы данных**

Для отображения информационной модели были выделены следующие сущности:

* Автомобили;
* Владельцы;
* Инспекторы;
* Заказы.

Сущность «Автомобили» хранит информацию об автомобилях, проходящих техническое обслуживание на СТО. Атрибуты сущности:

* Идентификатор;
* Номер;
* Модель;
* Объем двигателя;
* Номер кузова;
* Год производства.

Сущность «Владельцы» хранит информацию о владельцах автомобилей. Атрибуты данной сущности:

* Идентификатор;
* Имя;
* Фамилия;
* Отчество;
* Номер телефона.

Сущность «Инспекторы» хранит информацию о сотрудниках СТО, проводящих техническое обслуживание автомобилей. Атрибуты:

* Идентификатор;
* Имя;
* Фамилия;
* Отчество;
* Должность;
* Заработная плата.

Сущность «Заказы» хранит информацию о всех оформленных заказах. Атрибуты данной сущности:

* Идентификатор;
* Дата;
* Наименование услуги;
* Цена;
* Идентификатор автомобиля;
* Идентификатор владельца;
* Идентификатор инспектора;
* Отметка о выполнении.

Физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к данным. Построение физической модели проводилось на основе логической модели базы данных. Физическая модель представлена на рисунке 2.1.

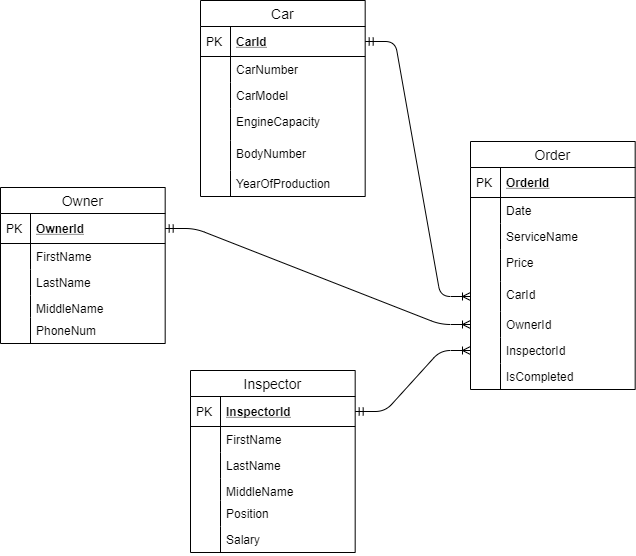


Рисунок 2.1 – Физическая модель базы данных

* 1. **Описание технологии решения поставленной задачи**

*ASP.NET (Active Server Pages для .NET)* – платформа разработки веб-приложений, в состав которой входит: веб-сервисы, программная инфраструктура, модель программирования, от компании Майкрософт. *ASP.NET* входит в состав платформы .*NET Framework* и является развитием более старой технологии *Microsoft ASP*.

*ASP.NET* внешне во многом сохраняет схожесть с более старой технологией *ASP*, что позволяет разработчикам относительно легко перейти на *ASP.NET*. В то же время внутреннее устройство *ASP.NET* существенно отличается от *ASP*, поскольку она основана на платформе *.NET* и, следовательно, использует все новые возможности, предоставляемые этой платформой.

Поскольку *ASP.NET* основывается на *Common Language Runtime (CLR),* которая является основой всех приложений *Microsoft .NET*, разработчики могут писать код для *ASP.NET*, используя языки программирования, входящие в комплект *.NET Framework (C#, Visual Basic.NET, J#* и *JScript .NET).*

Программная модель *ASP.NET* основывается на протоколе *HTTP* и использует его правила взаимодействия между сервером и браузером. При формировании страницы заложена абстрактная программная модель *Web Forms* и на ней основана основная часть реализации программного кода.

В *ASP.NET* поддерживается несколько программных моделей для создания веб-приложений:

* *ASP.NET Web Forms* – фреймворк для создания модульных веб-страниц из компонентов с обработкой событий пользовательского интерфейса на стороне сервера;
* *ASP.NET MVC* – фреймворк для создания веб-страниц с использованием шаблона проектирования *MVC*;
* *ASP.NET Web Pages* – упрощённый синтаксис для добавления динамического кода и доступа к данным внутри *HTML* разметки веб-страниц;
* *ASP.NET Web API* – фреймворк для создания *Web API* поверх *.NET Framework*;
* *ASP.NET WebHooks* – реализация шаблона *Webhook* для подписки на события и публикации событий через *HTTP*;
* *SignalR* — фреймворк для двунаправленного обмена сообщениями в реальном времени между клиентом и сервером.

Другие расширения *ASP.NET*:

* *ASP.NET Handler* – компоненты, реализующие интерфейс *System.Web.IHttpHandler*. В отличие от страниц *ASP.NET* у них нет файла с *HTML* разметкой, нет поддержки обработки событий и других вспомогательных технологий. Они содержат только файл с программным кодом, написанным на любом из .*NET*-совместимых языков, который записывает какие-то данные в *HTTP* ответ. *HTTP* обработчики схожи с *ISAPI*-расширениями;
* *ASP.NET AJAX* – расширение, содержащее как клиентские, так и серверные компоненты для создания *ASP.NET* страниц, реализующих *AJAX* функциональность;
* *ASP.NET Dynamic Data* – скаффолдинговое расширение для создания управляемых данными приложений.
* В *ASP.NET*, для управления системой аутентификации пользователей на сайте, был реализован за счет *Membership API*, который представляет средства для входа, хранения и управления учетными записями пользователей. Архитектура *Membership API* выстроена таким образом, что позволяет управлять пользователями хранящимся в различных источниках: *Microsoft SQL Server*, *Microsoft Active Directory* или специальное хранилище (реализованное самим, но тогда необходимо самому реализовывать поставщик, чтобы подключиться к обобщенной базе поставщиков). В *ASP.NET* аутентификация может осуществляется с помощью форм, либо с помощью *Windows* через *IIS*.

Преимущества *ASP.NET* перед *ASP*:

* Компилируемый код выполняется быстрее, большинство ошибок отлавливается ещё на стадии разработки;
* Значительно улучшенная обработка ошибок во время выполнения запущенной готовой программы, с использованием блоков *try..catch*;
* Пользовательские элементы управления (*controls*) позволяют выделять часто используемые шаблоны, такие как меню сайта;
* Использование метафор, уже применяющихся в *Windows*-приложениях, например, таких как элементы управления и события;
* Расширяемый набор элементов управления и библиотек классов позволяет быстрее разрабатывать приложения;
* *ASP.NET* опирается на многоязыковые возможности .*NET*, что позволяет писать код страниц на *VB.NET, Delphi.NET, Visual C#, J#* и т. д.;
* Возможность кэширования всей страницы или её части для увеличения производительности;
* Возможность кэширования данных, используемых на странице;
* Возможность разделения визуальной части и бизнес-логики по разным файлам («*code behind*»);
* Расширяемая модель обработки запросов;
* Расширенная событийная модель;
* Расширяемая модель серверных элементов управления;
* Наличие *master*-страниц для задания шаблонов оформления страниц;
* Поддержка *CRUD*-операций при работе с таблицами через *GridView*;
* Встроенная поддержка *AJAX* [4].
  1. **Описание алгоритма решения поставленной задачи**

В результате разработки приложения были разработаны три проекта: *BusinessLayer*, *DataLayer* и *ServiceStationWebApp*. Проекты представляют собой слои трехуровневой архитектуры приложения. Трёхуровневая архитектура (трёхзвенная архитектура) – архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

Таким образом, *ServiceStationWebApp* – интерфейсный компонент, предоставляемый пользователю (слой клиента). Он не имеет прямых связей с уровнем данных, может быть дополнен некоторой бизнес-логикой и хранить состояние приложения. На этот уровень обычно выносится только простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции с данными (сортировка, группировка, подсчёт значений), уже загруженными на терминал.

*BusinessLayer* (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры). Реализация данного компонента обеспечивается связующим программным обеспечением. Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

*DataLayer* (слой данных) обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

В простейших конфигурациях все компоненты или часть из них могут быть совмещены на одном вычислительном узле. В продуктивных конфигурациях как правило используется выделенный вычислительный узел для сервера баз данных или кластер серверов баз данных, для серверов приложений — выделенная группа вычислительных узлов, к которым непосредственно подключаются клиенты (терминалы).

Проект *ServiceStationWebApp* реализует паттерн *MVC (Model-View-Controller)* – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер таким образом, что модификация каждого из них может осуществляться независимо.

Модель представляет собой данные и реагирует на команды контроллера, изменяя свое состояние.

Представление отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер в свою очередь интерпретирует действия пользователя, оповещая модели о необходимости изменений.

* 1. **Структура приложения**

Приложение состоит из трех проектов, и условно его можно разделить на три слоя: данные, бизнес-логика и представление.

Общая структура классов представлена на рисунке 2.2. Код приложения описан в приложении А.

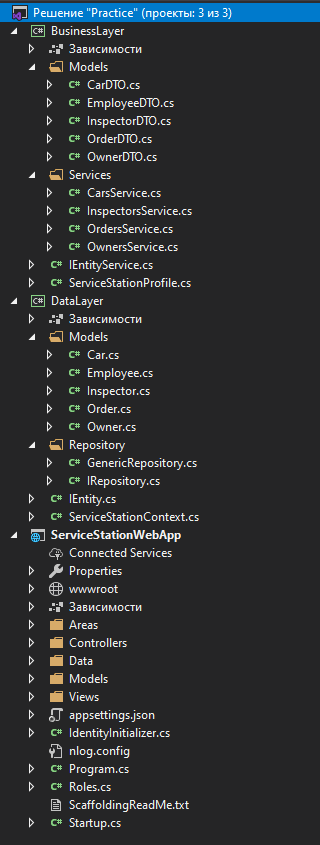


Рисунок 2.2 – Структура классов приложения

Схема программы представлена на рисунке 2.3.

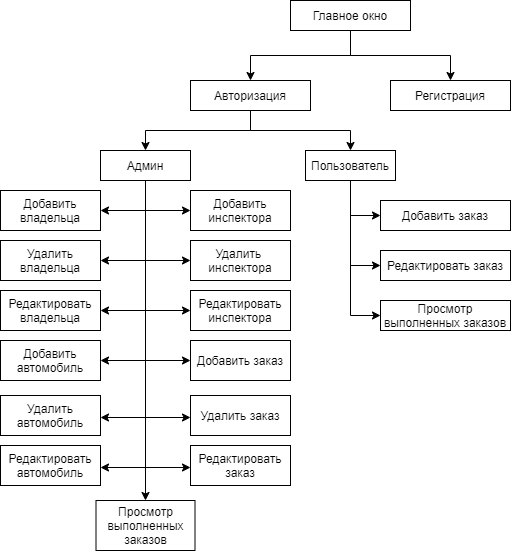


Рисунок 2.3 – Схема программы

* 1. **Общие сведения для работы с приложением**

При разработке приложения были использованы следующие инструменты:

* интегрированная среда разработки *Visual Studio* 2019;
* фреймворк для реализации *MVC*-паттерна *ASP.NET Core*.

Приложение предназначено для автоматизации работы администратора СТО, а именно *Web*-сайта с возможностью оформлять заказы, осуществлять управление справочниками и получать отчет о выполненных заказах.

* 1. **Описание интерфейса пользователя**

При запуске пользователь видит стартовую страницу приложения, представленное на рисунке 2.4.

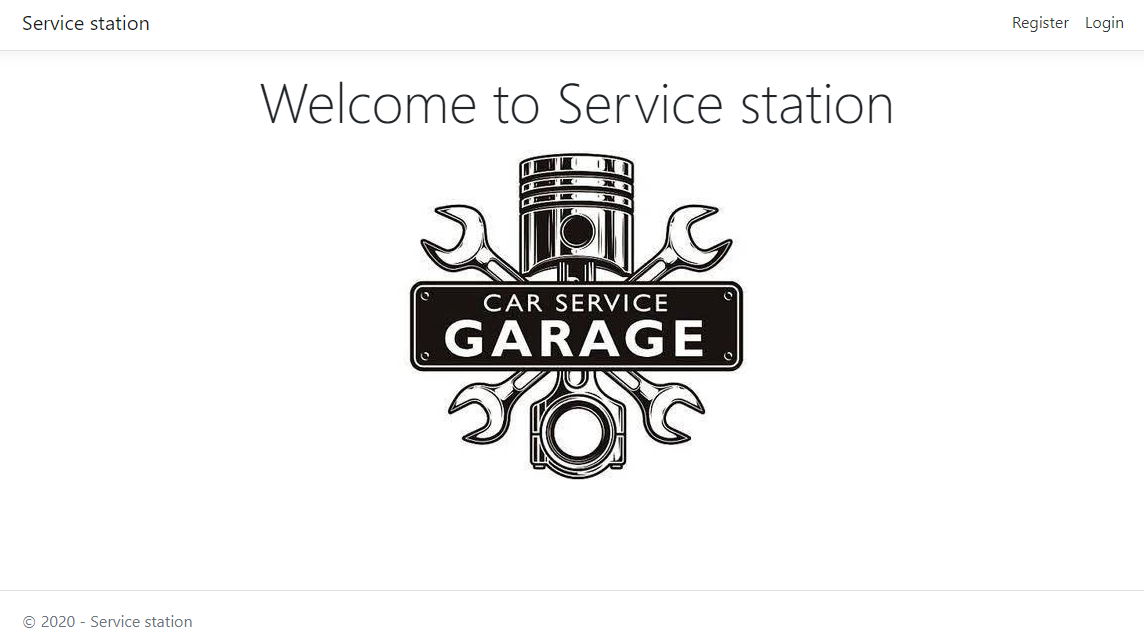


Рисунок 2.4 – Стартовая страница

На данной странице присутствуют кнопки для перехода на страницы регистрации и авторизации. При нажатии на кнопку *Register* открывается страница создания нового аккаунта, представленная на рисунке 2.5. Здесь пользователь вводит данные: почту и пароль, введенные данные проверяются и либо регистрация проходит успешно, и пользователь автоматически авторизуется (рисунок 2.6), либо получает уведомление об ошибке (рисунок 2.13). Также на странице присутствуют кнопка перехода на главную страницу и кнопка, ведущая на страницу авторизации (рисунок 2.7).

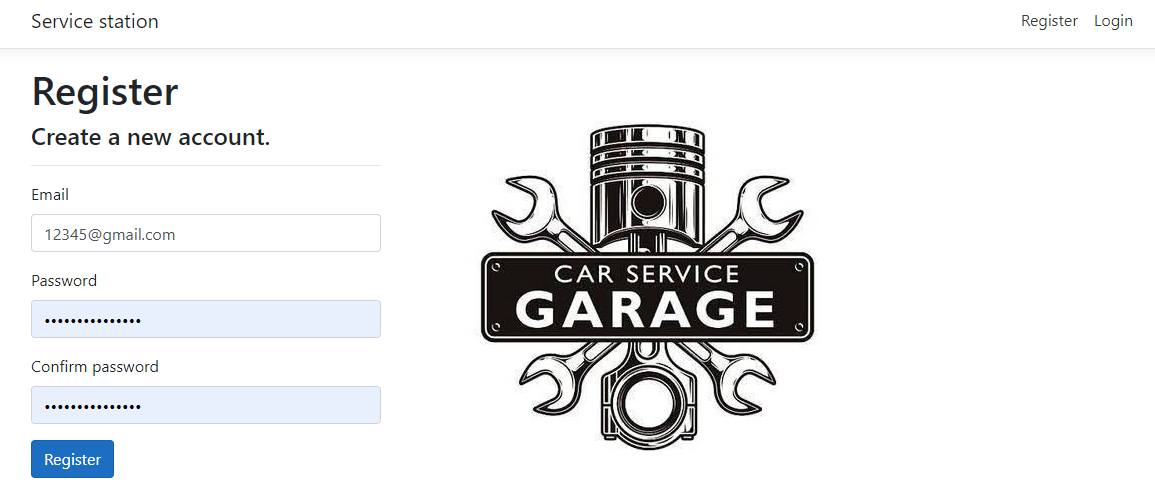


Рисунок 2.5 – Страница регистрации

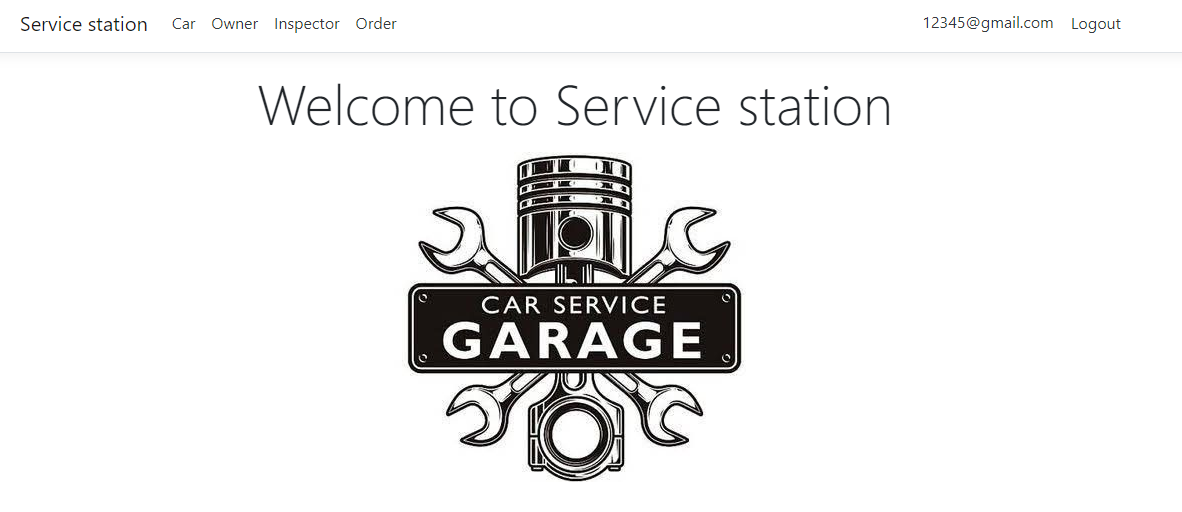


Рисунок 2.6 – Главная страница авторизованного пользователя

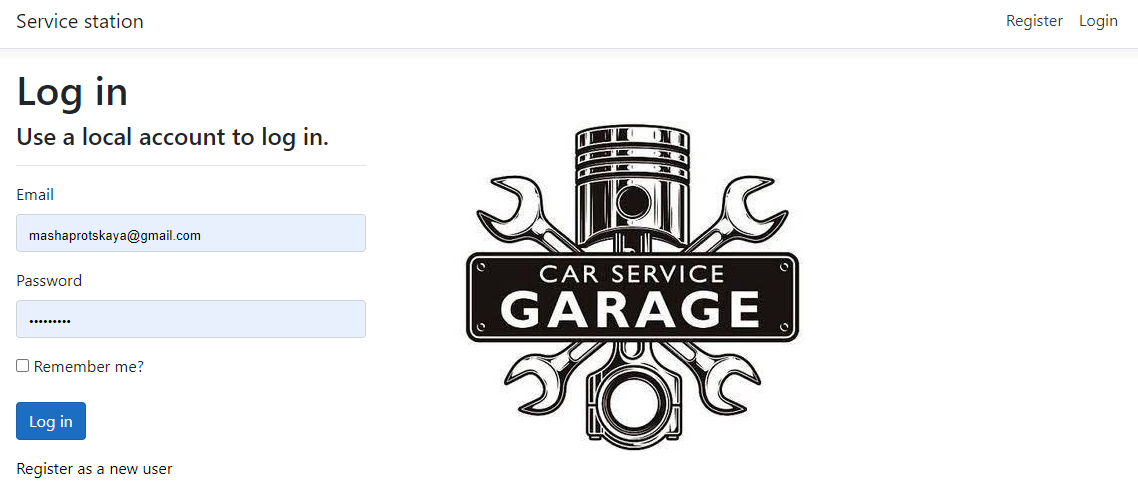


Рисунок 2.7 – Страница авторизации

Страница авторизации содержит кнопку перехода на страницу регистрации нового пользователя.

Главная страница авторизованного пользователя содержит кнопку для выхода.

В программе предусмотрены две роли: пользователь (администратор СТО) и админ. Админ может управлять всеми справочниками (рисунки 2.8-2.10), в то время как пользователь может только оформлять (редактировать) заказы, просматривать справочники (рисунок 2.11) и отчет о выполненных заказах (рисунок 2.12).

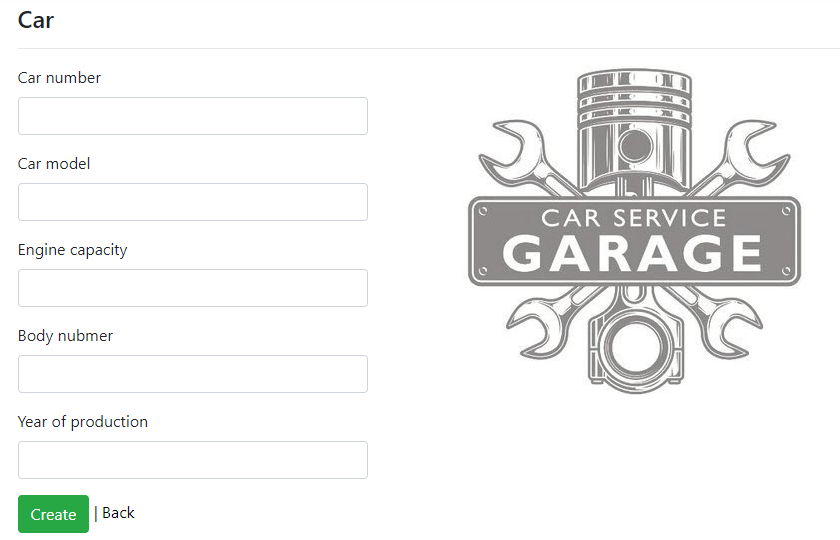


Рисунок 2.8 – Добавление нового автомобиля

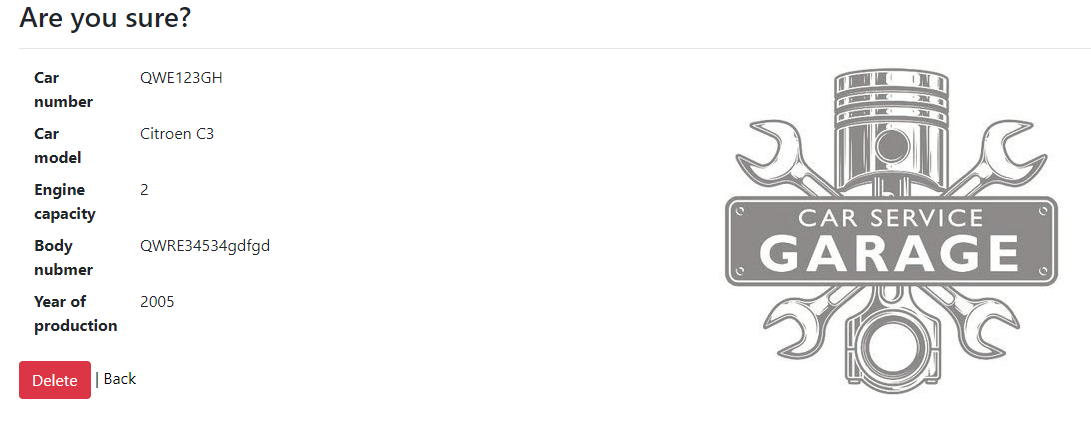


Рисунок 2.9 – Удаление автомобиля

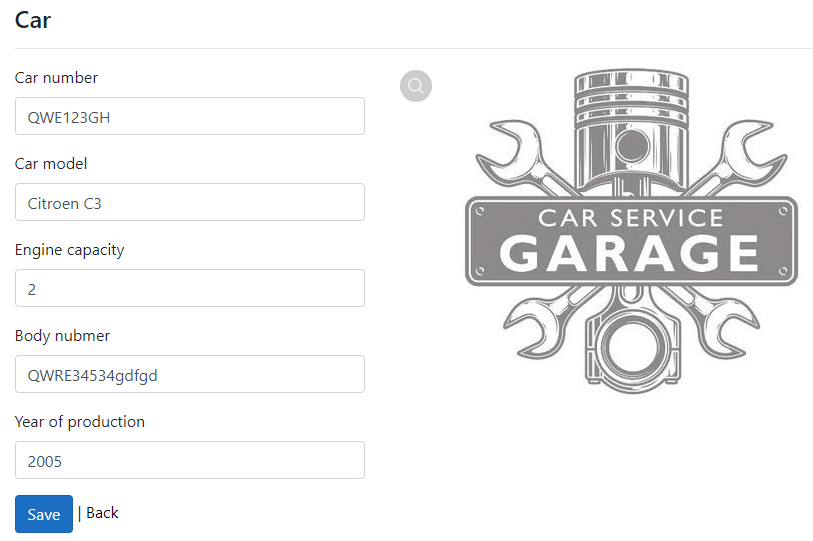


Рисунок 2.10 – Редактирование информации об автомобиле

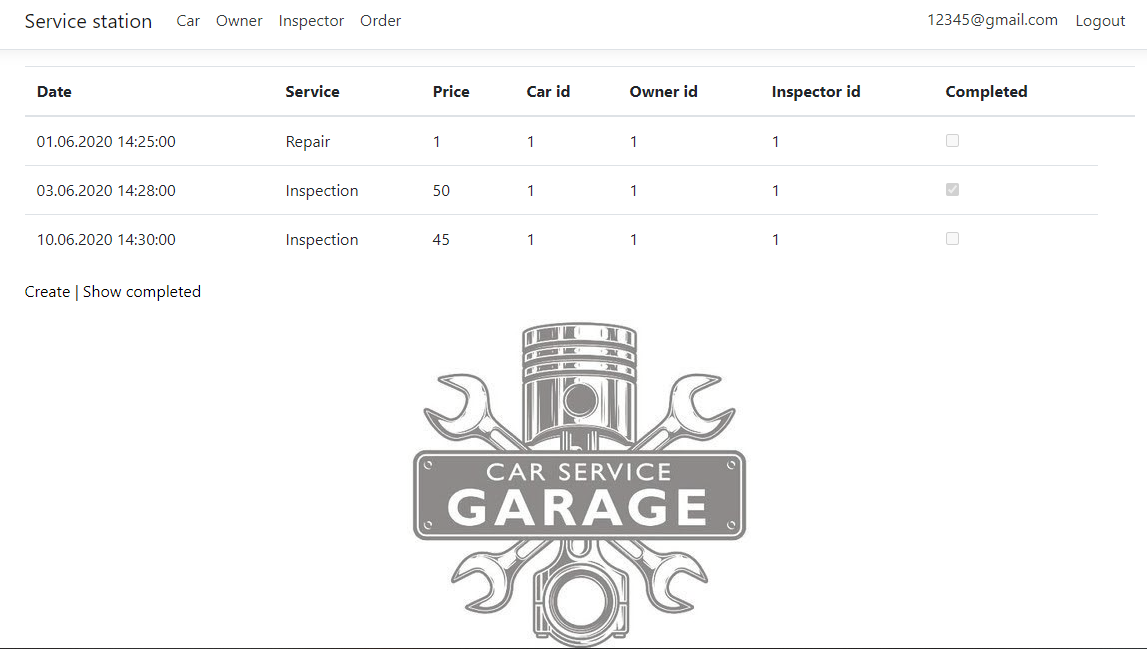


Рисунок 2.11 – Справочник – Заказы

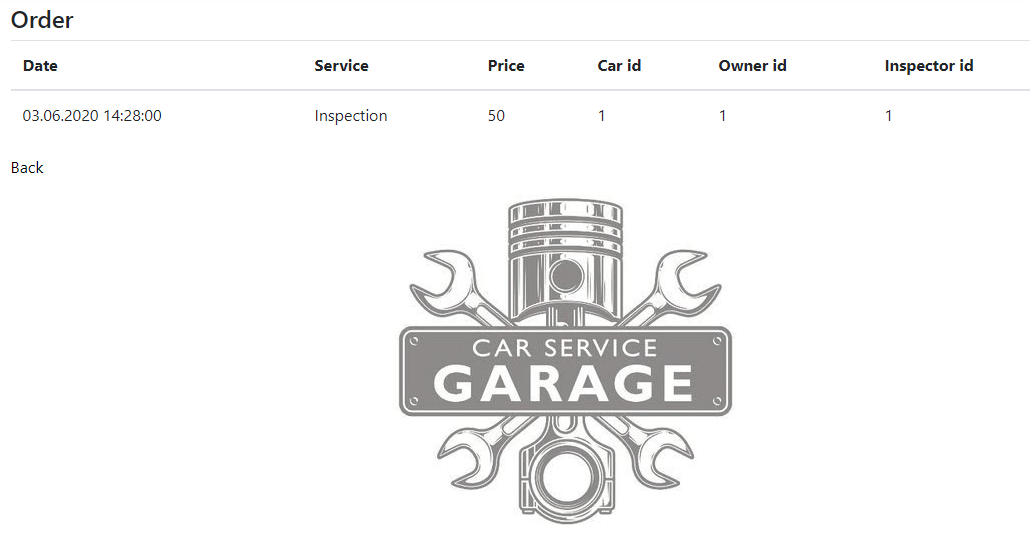


Рисунок 2.12 – Отчет о выполненных заказах

* 1. **Тестирование приложения**

Приложение было успешно протестировано на возникновение различных исключительных ситуаций.

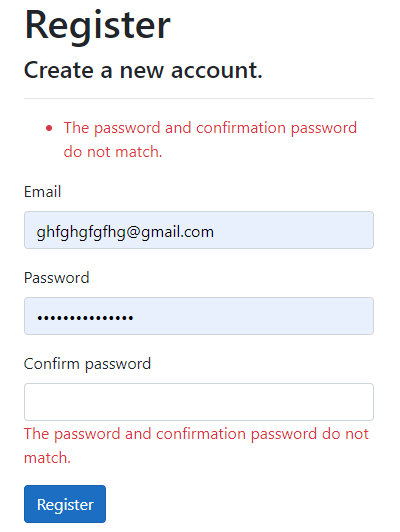


Рисунок 2.13 – Сообщение об ошибке при регистрации

При попытке оставить какие-либо поля пустыми при добавлении новой записи (редактировании существующей) пользователь получает соответствующее сообщение (рисунок 2.14).

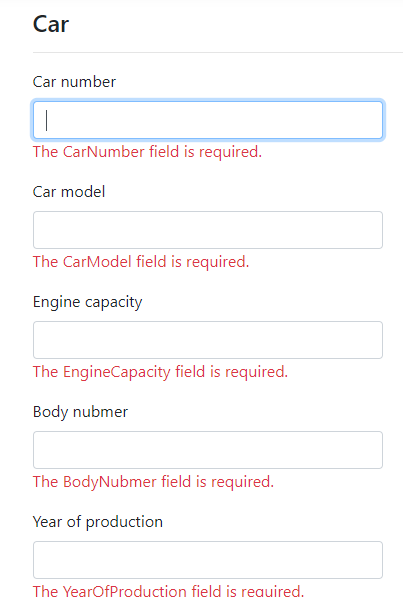


Рисунок 2.14 – Сообщение об ошибке при добавлении записи

При вводе неверных данных на странице авторизации пользователь также получит соответствующее сообщение (рисунок 2.15).

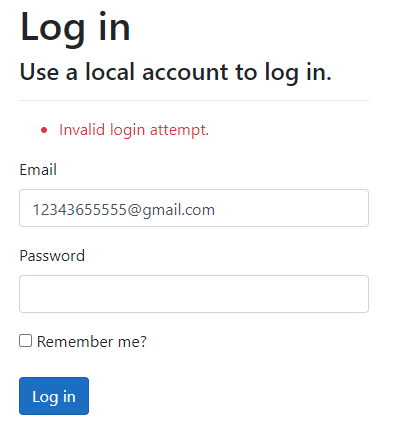


Рисунок 2.15 – Сообщение об ошибке авторизации

# Заключение

В процессе прохождения технологической(производственной) практики было разработано web-приложение на тему «Автоматизация СТО» на базе технологии *ASP*.

Были закреплены навыки работы с *ASP.NET Core*, паттерном *MVC*, разверткой приложения на *Linux*.

Также было выполнено индивидуальное задание, результатом которого стало web-приложение, в котором можно работать с различными справочниками, необходимыми на СТО, осуществлять управление пользователями, а также получать отчетные данные.

По итогам прохождения практики можно сделать вывод, что предприятие ООО «ИВА-Гомель-Парк» является поставщиком услуг в области разработки проектного программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд., – СПб.: Питер, 2014. – 816 с.
2. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд., – М.: Вильямс, 2013. – 1312 с.
3. Шилдт, Г. С# 4.0: Полное руководство – M.: Вильямс, 2010. – 1056 с.
4. Руководство по *ASP.NET Core* 3 – Электронн. данные. – Режим доступа: <https://metanit.com/sharp/aspnet5/> – Дата доступа: 25.06.2020.
5. Myfin.by. – Электронн. данные. – Режим доступа: <https://myfin.by/stati/view/13235-skolko-avto-prihoditsya-na-1000-belorusov>. – Дата доступа: 25.06.2020.
6. Кодексы Беларуси. – Электронн. данные. – Режим доступа: <http://pravo.levonevsky.org/bazaby/org335/basic/text0532.htm>. – Дата доступа: 25.06.2020.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**CarDTO.cs**

using DataLayer;

using System.ComponentModel.DataAnnotations;

namespace BusinessLayer.Models

{

public class CarDTO : IEntity

{

public int Id { get; set; }

[Required]

public string CarNumber { get; set; }

[Required]

public string CarModel { get; set; }

[Required]

public double EngineCapacity { get; set; }

[Required]

public string BodyNubmer { get; set; }

[Required]

public int YearOfProduction { get; set; }

public override string ToString()

{

return $"{Id} {CarNumber} {CarModel} {EngineCapacity} {BodyNubmer} {YearOfProduction}";

}

}

}

**EmployeeDTO.cs**

namespace BusinessLayer.Models

{

public class EmployeeDTO

{

public string Surname { get; set; }

public string FirstName { get; set; }

}

}

**InspectorDTO.cs**

using DataLayer;

using System.ComponentModel.DataAnnotations;

namespace BusinessLayer.Models

{

public class InspectorDTO : IEntity

{

public int Id { get; set; }

[Required]

public string Firstname { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string MiddleName { get; set; }

[Required]

public string Position { get; set; }

[Required]

public double Salary { get; set; }

public override string ToString()

{

return $"{Id} {Firstname} {LastName} {MiddleName} {Position} {Salary}";

}

}

}

**OrderDTO.cs**

using DataLayer;

using System;

using System.ComponentModel.DataAnnotations;

namespace BusinessLayer.Models

{

public class OrderDTO : IEntity

{

public int Id { get; set; }

[Required]

public DateTime Date { get; set; }

[Required]

public string ServiceName { get; set; }

[Required]

public int Price { get; set; }

[Required]

public int CarId { get; set; }

[Required]

public int OwnerId { get; set; }

[Required]

public int InspectorId { get; set; }

[Required]

public bool IsCompleted { get; set; }

public override string ToString()

{

return $"{Id} {Date} {ServiceName} {Price} {CarId} {OwnerId} {InspectorId} {IsCompleted}";

}

}

}

**OwnerDTO.cs**

using DataLayer;

using System.ComponentModel.DataAnnotations;

namespace BusinessLayer.Models

{

public class OwnerDTO : IEntity

{

public int Id { get; set; }

[Required]

public string Firstname { get; set; }

[Required]

public string LastName { get; set; }

[Required]

public string MiddleName { get; set; }

[Required]

public string PhoneNum { get; set; }

public override string ToString()

{

return $"{Id} {Firstname} {LastName} {MiddleName} {PhoneNum}";

}

}

}

**CarsService.cs**

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

using DataLayer.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace BusinessLayer.Services

{

public class CarsService : IEntityService<CarDTO>

{

private readonly IRepository<Car> \_carRepository;

private readonly IMapper \_mapper;

public CarsService(IRepository<Car> carRepository, IMapper mapper)

{

\_carRepository = carRepository;

\_mapper = mapper;

}

public async Task<CarDTO> GetItem(int id)

{

var item = await \_carRepository.GetById(id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

return \_mapper.Map<CarDTO>(item);

}

public async Task<IEnumerable<CarDTO>> GetItems()

{

return \_mapper.Map<IEnumerable<Car>, List<CarDTO>>(await \_carRepository.GetAll());

}

public async Task Create(CarDTO car)

{

if (car == null)

{

throw new InvalidOperationException(nameof(car));

}

else

{

await \_carRepository.Add(\_mapper.Map<Car>(car));

}

}

public async Task Update(CarDTO car)

{

if (car == null)

{

throw new InvalidOperationException(nameof(car));

}

else

{

await \_carRepository.Update(\_mapper.Map<Car>(car));

}

}

public async Task Delete(int id)

{

var item = (await GetItems()).Single(el => el.Id == id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_carRepository.Delete(\_mapper.Map<Car>(item).Id);

}

}

}

}

**InspectorsService.cs**

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

using DataLayer.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace BusinessLayer.Services

{

public class InspectorsService : IEntityService<InspectorDTO>

{

private readonly IRepository<Inspector> \_inspectorRepository;

private readonly IMapper \_mapper;

public InspectorsService(IRepository<Inspector> inspectorRepository, IMapper mapper)

{

\_inspectorRepository = inspectorRepository;

\_mapper = mapper;

}

public async Task<InspectorDTO> GetItem(int id)

{

var item = await \_inspectorRepository.GetById(id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

return \_mapper.Map<InspectorDTO>(item);

}

public async Task<IEnumerable<InspectorDTO>> GetItems()

{

return \_mapper.Map<IEnumerable<Inspector>, List<InspectorDTO>>(await \_inspectorRepository.GetAll());

}

public async Task Create(InspectorDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_inspectorRepository.Add(\_mapper.Map<Inspector>(item));

}

}

public async Task Update(InspectorDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_inspectorRepository.Update(\_mapper.Map<Inspector>(item));

}

}

public async Task Delete(int id)

{

var item = (await GetItems()).Single(el => el.Id == id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_inspectorRepository.Delete(\_mapper.Map<Inspector>(item).Id);

}

}

}

}

**OrdersService.cs**

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

using DataLayer.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace BusinessLayer.Services

{

public class OrdersService : IEntityService<OrderDTO>

{

private readonly IRepository<Order> \_orderRepository;

private readonly IRepository<Car> \_carRepository;

private readonly IRepository<Inspector> \_inspectorRepository;

private readonly IRepository<Owner> \_ownerRepository;

private readonly IMapper \_mapper;

public OrdersService(IRepository<Order> conditionRepository, IRepository<Car> carRepository,

IRepository<Inspector> inspectorRepository, IRepository<Owner> ownerRepository ,IMapper mapper)

{

\_orderRepository = conditionRepository;

\_carRepository = carRepository;

\_inspectorRepository = inspectorRepository;

\_ownerRepository = ownerRepository;

\_mapper = mapper;

}

public async Task<OrderDTO> GetItem(int id)

{

var item = await \_orderRepository.GetById(id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

return \_mapper.Map<OrderDTO>(item);

}

public async Task<IEnumerable<OrderDTO>> GetItems()

{

return \_mapper.Map<IEnumerable<Order>, List<OrderDTO>>(await \_orderRepository.GetAll());

}

public async Task Create(OrderDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

try

{

await \_inspectorRepository.GetById(item.InspectorId);

await \_carRepository.GetById(item.CarId);

await \_ownerRepository.GetById(item.OwnerId);

}

catch

{

throw new InvalidOperationException();

}

await \_orderRepository.Add(\_mapper.Map<Order>(item));

}

}

public async Task Update(OrderDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

try

{

await \_inspectorRepository.GetById(item.InspectorId);

await \_carRepository.GetById(item.CarId);

await \_ownerRepository.GetById(item.OwnerId);

}

catch

{

throw new InvalidOperationException();

}

await \_orderRepository.Update(\_mapper.Map<Order>(item));

}

}

public async Task Delete(int id)

{

var item = (await GetItems()).Single(el => el.Id == id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_orderRepository.Delete(\_mapper.Map<Order>(item).Id);

}

}

}

}

**OwnersService.cs**

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

using DataLayer.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace BusinessLayer.Services

{

public class OwnersService : IEntityService<OwnerDTO>

{

private readonly IRepository<Owner> \_ownerRepository;

private readonly IMapper \_mapper;

public OwnersService(IRepository<Owner> ownerRepository, IMapper mapper)

{

\_ownerRepository = ownerRepository;

\_mapper = mapper;

}

public async Task<OwnerDTO> GetItem(int id)

{

var item = await \_ownerRepository.GetById(id);

if(item == null)

{

throw new InvalidOperationException(nameof(item));

}

return \_mapper.Map<OwnerDTO>(item);

}

public async Task<IEnumerable<OwnerDTO>> GetItems()

{

return \_mapper.Map<IEnumerable<Owner>, List<OwnerDTO>>(await \_ownerRepository.GetAll());

}

public async Task Create(OwnerDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_ownerRepository.Add(\_mapper.Map<Owner>(item));

}

}

public async Task Update(OwnerDTO item)

{

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_ownerRepository.Update(\_mapper.Map<Owner>(item));

}

}

public async Task Delete(int id)

{

var item = (await GetItems()).Single(el => el.Id == id);

if (item == null)

{

throw new InvalidOperationException(nameof(item));

}

else

{

await \_ownerRepository.Delete(\_mapper.Map<Owner>(item).Id);

}

}

}

}

**IEntityService.cs**

using System.Collections.Generic;

using System.Threading.Tasks;

namespace BusinessLayer

{

public interface IEntityService<T> where T : class

{

Task Create(T item);

Task Delete(int id);

Task Update(T item);

Task<T> GetItem(int id);

Task<IEnumerable<T>> GetItems();

}

}

ServiceStationProfile.cs

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

namespace BusinessLayer

{

public class ServiceStationProfile : Profile

{

public ServiceStationProfile()

{

CreateMap<Car, CarDTO>().ReverseMap();

CreateMap<Order, OrderDTO>().ReverseMap();

CreateMap<Inspector, InspectorDTO>().ReverseMap();

CreateMap<Owner, OwnerDTO>().ReverseMap();

CreateMap<Employee, EmployeeDTO>().ReverseMap();

}

}

}

**Car.cs**

using AutoMapper;

using BusinessLayer.Models;

using DataLayer.Models;

namespace BusinessLayer

{

public class ServiceStationProfile : Profile

{

public ServiceStationProfile()

{

CreateMap<Car, CarDTO>().ReverseMap();

CreateMap<Order, OrderDTO>().ReverseMap();

CreateMap<Inspector, InspectorDTO>().ReverseMap();

CreateMap<Owner, OwnerDTO>().ReverseMap();

CreateMap<Employee, EmployeeDTO>().ReverseMap();

}

}

}

**Employee.cs**

using Microsoft.AspNetCore.Identity;

namespace DataLayer.Models

{

public class Employee : IdentityUser

{

public string Surname { get; set; }

public string FirstName { get; set; }

}

}

**Inspector.cs**

namespace DataLayer.Models

{

public class Inspector : IEntity

{

public int Id { get; set; }

public string Firstname { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public string Position { get; set; }

public double Salary { get; set; }

}

}

**Order.cs**

using System;

namespace DataLayer.Models

{

public class Order : IEntity

{

public int Id { get; set; }

public DateTime Date { get; set; }

public string ServiceName { get; set; }

public int Price { get; set; }

public int CarId { get; set; }

public int OwnerId { get; set; }

public int InspectorId { get; set; }

public bool IsCompleted { get; set; }

}

}

**Owner.cs**

namespace DataLayer.Models

{

public class Owner : IEntity

{

public int Id { get; set; }

public string Firstname { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public string PhoneNum { get; set; }

}

}

**GenericRepository.cs**

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace DataLayer.Repository

{

public class GenericRepository<T> : IRepository<T> where T : class, IEntity

{

private readonly ServiceStationContext serviceStationContext;

public GenericRepository(ServiceStationContext context)

{

serviceStationContext = context;

}

public async Task Add(T item)

{

await serviceStationContext.Set<T>().AddAsync(item);

await serviceStationContext.SaveChangesAsync();

}

public async Task Delete(int id)

{

var item = serviceStationContext.Set<T>().Find(id);

if (item != null)

{

serviceStationContext.Set<T>().Remove(item);

}

await serviceStationContext.SaveChangesAsync();

}

public async Task Update(T item)

{

var entry = await serviceStationContext.Set<T>().FirstAsync(e => e.Id == item.Id);

serviceStationContext.Entry(entry).CurrentValues.SetValues(item);

await serviceStationContext.SaveChangesAsync();

}

public async Task<IEnumerable<T>> GetAll()

{

return await serviceStationContext.Set<T>().ToListAsync();

}

public async Task<T> GetById(int id)

{

return await serviceStationContext.Set<T>().FindAsync(id);

}

}

}

**IRepository.cs**

using System.Collections.Generic;

using System.Threading.Tasks;

namespace DataLayer.Repository

{

public interface IRepository<T> where T : class, IEntity

{

Task Add(T item);

Task Update(T item);

Task Delete(int id);

Task<IEnumerable<T>> GetAll();

Task<T> GetById(int id);

}

}

**IEntity.cs**

namespace DataLayer

{

public interface IEntity

{

int Id { get; }

}

}

ServiceStationContext.cs

using DataLayer.Models;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace DataLayer

{

public class ServiceStationContext : IdentityDbContext<Employee>

{

public DbSet<Car> Cars { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<Inspector> Inspectors { get; set; }

public DbSet<Owner> Owners { get; set; }

public ServiceStationContext(DbContextOptions<ServiceStationContext> options)

: base(options)

{

Database.EnsureCreated();

}

}

}

**Login.cshtml**

@page

@model LoginModel

@{

ViewData["Title"] = "Log in";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-4">

<section>

<form id="account" method="post">

<h4>Use a local account to log in.</h4>

<hr />

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

<div class="form-group">

<label **asp-for**="Input.Email"></label>

<input **asp-for**="Input.Email" class="form-control" />

<span **asp-validation-for**="Input.Email" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Input.Password"></label>

<input **asp-for**="Input.Password" class="form-control" />

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

</div>

<div class="form-group">

<div class="checkbox">

<label **asp-for**="Input.RememberMe">

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

@Html.DisplayNameFor(m => m.Input.RememberMe)

</label>

</div>

</div>

<div class="form-group">

<button type="submit" class="btn btn-primary">Log in</button>

</div>

<div class="form-group">

<p>

<a **asp-page**="./Register" **asp-route-returnUrl**="@Model.ReturnUrl">Register as a new user</a>

</p>

</div>

</form>

</section>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" />

</div>

</div>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**Login.cshtml.cs**

using DataLayer.Models;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.Extensions.Logging;

using System.ComponentModel.DataAnnotations;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Areas.Identity.Pages.Account

{

[AllowAnonymous]

public class LoginModel : PageModel

{

private readonly SignInManager<Employee> \_signInManager;

private readonly ILogger<LoginModel> \_logger;

public LoginModel(SignInManager<Employee> signInManager,

ILogger<LoginModel> logger)

{

\_signInManager = signInManager;

\_logger = logger;

}

[BindProperty]

public InputModel Input { get; set; }

public string ReturnUrl { get; set; }

[TempData]

public string ErrorMessage { get; set; }

public class InputModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Remember me?")]

public bool RememberMe { get; set; }

}

public async Task OnGetAsync(string returnUrl = null)

{

if (!string.IsNullOrEmpty(ErrorMessage))

{

ModelState.AddModelError(string.Empty, ErrorMessage);

}

returnUrl = returnUrl ?? Url.Content("~/");

await HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);

ReturnUrl = returnUrl;

}

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

returnUrl = returnUrl ?? Url.Content("~/");

if (ModelState.IsValid)

{

var result = await \_signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: false);

if (result.Succeeded)

{

\_logger.LogInformation("User logged in.");

return LocalRedirect(returnUrl);

}

else

{

ModelState.AddModelError(string.Empty, "Invalid login attempt.");

return Page();

}

}

return Page();

}

}

}

**Logout.cshtml**

@page

@model LogoutModel

@{

ViewData["Title"] = "Log out";

}

<header>

<h1>@ViewData["Title"]</h1>

<p>You have successfully logged out of the application.</p>

</header>

**Logout.cshtml.cs**

using DataLayer.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.Extensions.Logging;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Areas.Identity.Pages.Account

{

[AllowAnonymous]

public class LogoutModel : PageModel

{

private readonly SignInManager<Employee> \_signInManager;

private readonly ILogger<LogoutModel> \_logger;

public LogoutModel(SignInManager<Employee> signInManager, ILogger<LogoutModel> logger)

{

\_signInManager = signInManager;

\_logger = logger;

}

public void OnGet()

{

}

public async Task<IActionResult> OnPost(string returnUrl = null)

{

await \_signInManager.SignOutAsync();

\_logger.LogInformation("User logged out.");

if (returnUrl != null)

{

return LocalRedirect(returnUrl);

}

else

{

return RedirectToPage();

}

}

}

}

**Register.cshtml**

@page

@model RegisterModel

@{

ViewData["Title"] = "Register";

}

<h1>@ViewData["Title"]</h1>

<div class="row">

<div class="col-md-4">

<form **asp-route-returnUrl**="@Model.ReturnUrl" method="post">

<h4>Create a new account.</h4>

<hr />

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

<div class="form-group">

<label **asp-for**="Input.Email"></label>

<input **asp-for**="Input.Email" class="form-control" />

<span **asp-validation-for**="Input.Email" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Input.Password"></label>

<input **asp-for**="Input.Password" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Input.ConfirmPassword"></label>

<input **asp-for**="Input.ConfirmPassword" class="form-control" />

<span **asp-validation-for**="Input.ConfirmPassword" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Register</button>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" />

</div>

</div>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**Register.cshtml.cs**

using DataLayer.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.Extensions.Logging;

using System.ComponentModel.DataAnnotations;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Areas.Identity.Pages.Account

{

[AllowAnonymous]

public class RegisterModel : PageModel

{

private readonly SignInManager<Employee> \_signInManager;

private readonly UserManager<Employee> \_userManager;

private readonly ILogger<RegisterModel> \_logger;

public RegisterModel(

UserManager<Employee> userManager,

SignInManager<Employee> signInManager,

ILogger<RegisterModel> logger)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_logger = logger;

}

[BindProperty]

public InputModel Input { get; set; }

public string ReturnUrl { get; set; }

public class InputModel

{

[Required]

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm password")]

[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

}

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

returnUrl = returnUrl ?? Url.Content("~/");

if (ModelState.IsValid)

{

var user = new Employee { UserName = Input.Email, Email = Input.Email };

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

if (result.Succeeded)

{

\_logger.LogInformation("User created a new account with password.");

if (\_userManager.Options.SignIn.RequireConfirmedAccount)

{

return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl });

}

else

{

await \_signInManager.SignInAsync(user, isPersistent: false);

return LocalRedirect(returnUrl);

}

}

foreach (var error in result.Errors)

{

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

}

}

return Page();

}

}

}

**CarsController.cs**

using BusinessLayer.Models;

using BusinessLayer.Services;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Controllers

{

public class CarsController : Controller

{

private readonly CarsService \_carService;

private readonly ILogger<CarsController> \_logger;

public CarsController(CarsService carService, ILogger<CarsController> logger)

{

\_carService = carService;

\_logger = logger;

}

public async Task<IActionResult> Index()

{

return View(await \_carService.GetItems());

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(IFormCollection collection)

{

try

{

var car = new CarDTO

{

CarNumber = collection["CarNumber"],

CarModel = collection["CarModel"],

EngineCapacity = Convert.ToDouble(collection["EngineCapacity"]),

BodyNubmer = collection["BodyNumber"],

YearOfProduction = Convert.ToInt32(collection["YearOfProduction"])

};

await \_carService.Create(car);

\_logger.LogInformation("Creation was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Creation failed.", ex);

return View();

}

}

public async Task<IActionResult> Edit(int id)

{

return View(await \_carService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

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

{

try

{

var car = new CarDTO

{

Id = id,

CarNumber = collection["CarNumber"],

CarModel = collection["CarModel"],

EngineCapacity = Convert.ToDouble(collection["EngineCapacity"]),

BodyNubmer = collection["BodyNumber"],

YearOfProduction = Convert.ToInt32(collection["YearOfProduction"]),

};

await \_carService.Update(car);

\_logger.LogInformation("Editing was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Editing failed.", ex);

return View();

}

}

public async Task<IActionResult> Delete(int id)

{

return View(await \_carService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteById(int id)

{

try

{

await \_carService.Delete(id);

\_logger.LogInformation("Delete was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Delete failed.", ex);

return View();

}

}

}

}

**InspectorsController.cs**

using BusinessLayer.Models;

using BusinessLayer.Services;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Controllers

{

public class InspectorsController : Controller

{

private readonly InspectorsService \_inspectorService;

private readonly ILogger<InspectorsController> \_logger;

public InspectorsController(InspectorsService inspectorService, ILogger<InspectorsController> logger)

{

\_inspectorService = inspectorService;

\_logger = logger;

}

public async Task<IActionResult> Index()

{

return View(await \_inspectorService.GetItems());

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(IFormCollection collection)

{

try

{

var inspector = new InspectorDTO

{

Firstname = collection["Firstname"],

LastName = collection["LastName"],

MiddleName = collection["MiddleName"],

Position = collection["Position"],

Salary = Convert.ToDouble(collection["Salary"])

};

await \_inspectorService.Create(inspector);

\_logger.LogInformation("Creation was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Creation failed.", ex);

return View();

}

}

public async Task<IActionResult> Edit(int id)

{

return View(await \_inspectorService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

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

{

try

{

var owner = new InspectorDTO

{

Id = id,

Firstname = collection["Firstname"],

LastName = collection["LastName"],

MiddleName = collection["MiddleName"],

Position = collection["Position"],

Salary = Convert.ToDouble(collection["Salary"])

};

await \_inspectorService.Update(owner);

\_logger.LogInformation("Editing was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Editing failed.", ex);

return View();

}

}

public async Task<IActionResult> Delete(int id)

{

return View(await \_inspectorService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteById(int id)

{

try

{

await \_inspectorService.Delete(id);

\_logger.LogInformation("Delete was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Delete failed.", ex);

return View();

}

}

}

}

**OrdersController.cs**

using BusinessLayer.Models;

using BusinessLayer.Services;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Controllers

{

public class OrdersController : Controller

{

private readonly OrdersService \_orderService;

private readonly ILogger<OrdersController> \_logger;

public OrdersController(OrdersService orderService, ILogger<OrdersController> logger)

{

\_orderService = orderService;

\_logger = logger;

}

public async Task<IActionResult> Index()

{

return View(await \_orderService.GetItems());

}

public async Task<IActionResult> Completed()

{

return View(await \_orderService.GetItems());

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(IFormCollection collection)

{

try

{

var item = new OrderDTO

{

Date = Convert.ToDateTime(collection["Date"]),

ServiceName = collection["ServiceName"],

Price = Convert.ToInt32(collection["Price"]),

CarId = Convert.ToInt32(collection["CarId"]),

OwnerId = Convert.ToInt32(collection["OwnerId"]),

InspectorId = Convert.ToInt32(collection["InspectorId"]),

IsCompleted = Convert.ToBoolean(collection["IsCompleted"])

};

await \_orderService.Create(item);

\_logger.LogInformation("Creation was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Creation failed.", ex);

return View();

}

}

public async Task<IActionResult> Edit(int id)

{

return View(await \_orderService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

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

{

try

{

var item = new OrderDTO

{

Id = id,

Date = Convert.ToDateTime(collection["Date"]),

ServiceName = collection["ServiceName"],

Price = Convert.ToInt32(collection["Price"]),

CarId = Convert.ToInt32(collection["CarId"]),

OwnerId = Convert.ToInt32(collection["OwnerId"]),

InspectorId = Convert.ToInt32(collection["InspectorId"]),

IsCompleted = Convert.ToBoolean(collection["IsCompleted"])

};

await \_orderService.Update(item);

\_logger.LogInformation("Editing was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Editing failed.", ex);

return View();

}

}

public async Task<IActionResult> Delete(int id)

{

return View(await \_orderService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteById(int id)

{

try

{

await \_orderService.Delete(id);

\_logger.LogInformation("Delete was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Delete failed.", ex);

return View();

}

}

}

}

**OwnersController.cs**

using BusinessLayer.Models;

using BusinessLayer.Services;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System;

using System.Threading.Tasks;

namespace ServiceStationWebApp.Controllers

{

public class OwnersController : Controller

{

private readonly OwnersService \_ownerService;

private readonly ILogger<OwnersController> \_logger;

public OwnersController(OwnersService inspectorService, ILogger<OwnersController> logger)

{

\_ownerService = inspectorService;

\_logger = logger;

}

public async Task<IActionResult> Index()

{

return View(await \_ownerService.GetItems());

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(IFormCollection collection)

{

try

{

var owner = new OwnerDTO

{

Firstname = collection["Firstname"],

LastName = collection["LastName"],

MiddleName = collection["MiddleName"],

PhoneNum = collection["PhoneNum"]

};

await \_ownerService.Create(owner);

\_logger.LogInformation("Creation was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Creation failed.", ex);

return View();

}

}

public async Task<IActionResult> Edit(int id)

{

return View(await \_ownerService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

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

{

try

{

var owner = new OwnerDTO

{

Id = id,

Firstname = collection["Firstname"],

LastName = collection["LastName"],

MiddleName = collection["MiddleName"],

PhoneNum = collection["PhoneNum"]

};

await \_ownerService.Update(owner);

\_logger.LogInformation("Eediting was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Editing failed.", ex);

return View();

}

}

public async Task<IActionResult> Delete(int id)

{

return View(await \_ownerService.GetItem(id));

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteById(int id)

{

try

{

await \_ownerService.Delete(id);

\_logger.LogInformation("Delete was successful.");

return RedirectToAction(nameof(Index));

}

catch (Exception ex)

{

\_logger.LogError("Delete failed.", ex);

return View();

}

}

}

}

**Cars/Create.cshtml**

@model BusinessLayer.Models.CarDTO

<h4>Car</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Create">

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

<div class="form-group">

<label **asp-for**="CarNumber" class="control-label">Car number</label>

<input **asp-for**="CarNumber" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="CarModel" class="control-label">Car model</label>

<input **asp-for**="CarModel" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="EngineCapacity" class="control-label">Engine capacity</label>

<input **asp-for**="EngineCapacity" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="BodyNubmer" class="control-label">Body nubmer</label>

<input **asp-for**="BodyNubmer" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="YearOfProduction" class="control-label">Year of production</label>

<input **asp-for**="YearOfProduction" class="form-control" />

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

</div>

<div class="form-group">

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

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Cars/Delete.cshtml**

@model BusinessLayer.Models.CarDTO

<h3>Are you sure?</h3>

<div>

<hr />

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="float:right; opacity:0.5" />

<dl class="row">

<dt class="col-sm-2">

<label **asp-for**="CarNumber">Car number</label>

</dt>

<dd class="col-sm-10">

@Model.CarNumber

</dd>

<dt class="col-sm-2">

<label **asp-for**="CarModel">Car model</label>

</dt>

<dd class="col-sm-10">

@Model.CarModel

</dd>

<dt class="col-sm-2">

<label **asp-for**="EngineCapacity">Engine capacity</label>

</dt>

<dd class="col-sm-10">

@Model.EngineCapacity

</dd>

<dt class="col-sm-2">

<label **asp-for**="BodyNubmer">Body nubmer</label>

</dt>

<dd class="col-sm-10">

@Model.BodyNubmer

</dd>

<dt class="col-sm-2">

<label **asp-for**="YearOfProduction">Year of production</label>

</dt>

<dd class="col-sm-10">

@Model.YearOfProduction

</dd>

</dl>

<form **asp-action**="DeleteById">

<input **type**="hidden" **asp-for**="Id" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a **asp-action**="Index">Back</a>

</form>

</div>

**Cars/Edit.cshtml**

@model BusinessLayer.Models.CarDTO

<h4>Car</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Edit">

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

<input **type**="hidden" **asp-for**="Id" />

<div class="form-group">

<label **asp-for**="CarNumber" class="control-label">Car number</label>

<input **asp-for**="CarNumber" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="CarModel" class="control-label">Car model</label>

<input **asp-for**="CarModel" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="EngineCapacity" class="control-label">Engine capacity</label>

<input **asp-for**="EngineCapacity" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="BodyNubmer" class="control-label">Body nubmer</label>

<input **asp-for**="BodyNubmer" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="YearOfProduction" class="control-label">Year of production</label>

<input **asp-for**="YearOfProduction" class="form-control" />

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

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" /> |

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Cars/Index.cshtml**

@model IEnumerable<BusinessLayer.Models.CarDTO>

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

@using Microsoft.AspNetCore.Identity

<table class="table">

<thead>

<tr>

<th>

Car number

</th>

<th>

Car model

</th>

<th>

Engine capacity

</th>

<th>

Body nubmer

</th>

<th>

Year of production

</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.CarNumber)</td>

<td> @Html.DisplayFor(modelItem => item.CarModel)</td>

<td> @Html.DisplayFor(modelItem => item.EngineCapacity)</td>

<td> @Html.DisplayFor(modelItem => item.BodyNubmer)</td>

<td> @Html.DisplayFor(modelItem => item.YearOfProduction)</td>

@if (SignInManager.IsSignedIn(User) && User.IsInRole("admin"))

{

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id">Edit</a> |

<a **asp-action**="Delete" **asp-route-id**="@item.Id">Delete</a>

</td>

}

</tr>

}

</tbody>

</table>

<p>

<a **asp-action**="Create">Create</a>

</p>

<center>

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</center>

**Inspectors/Create.cshtml**

@model BusinessLayer.Models.InspectorDTO

<h4>Inspector</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Create">

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

<div class="form-group">

<label **asp-for**="Firstname" class="control-label">First name</label>

<input **asp-for**="Firstname" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label">Last name</label>

<input **asp-for**="LastName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="MiddleName" class="control-label">Middle name</label>

<input **asp-for**="MiddleName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Position" class="control-label">Position</label>

<input **asp-for**="Position" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Salary" class="control-label">Salary</label>

<input **asp-for**="Salary" class="form-control" />

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

</div>

<div class="form-group">

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

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Inspectors /Delete.cshtml**

@model BusinessLayer.Models.InspectorDTO

<h3>Are you sure?</h3>

<div>

<hr />

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="float:right; opacity:0.5" />

<dl class="row">

<dt class="col-sm-2">

<label **asp-for**="Firstname">First name</label>

</dt>

<dd class="col-sm-10">

@Model.Firstname

</dd>

<dt class="col-sm-2">

<label **asp-for**="LastName">Last name</label>

</dt>

<dd class="col-sm-10">

@Model.LastName

</dd>

<dt class="col-sm-2">

<label **asp-for**="MiddleName">Middle name</label>

</dt>

<dd class="col-sm-10">

@Model.MiddleName

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Model.Position

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Model.Salary

</dd>

</dl>

<form **asp-action**="DeleteById">

<input **type**="hidden" **asp-for**="Id" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a **asp-action**="Index">Back</a>

</form>

</div>

**Inspectors / Edit.cshtml**

@model BusinessLayer.Models.InspectorDTO

<h4>Inspector</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Edit">

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

<input **type**="hidden" **asp-for**="Id" />

<div class="form-group">

<label **asp-for**="Firstname" class="control-label">First name</label>

<input **asp-for**="Firstname" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label">Last name</label>

<input **asp-for**="LastName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="MiddleName" class="control-label">Middle name</label>

<input **asp-for**="MiddleName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Position" class="control-label">Position</label>

<input **asp-for**="Position" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Salary" class="control-label">Salary</label>

<input **asp-for**="Salary" class="form-control" />

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

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" /> |

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Inspectors / Index.cshtml**

@model IEnumerable<BusinessLayer.Models.InspectorDTO>

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

@using Microsoft.AspNetCore.Identity

<table class="table">

<thead>

<tr>

<th>

First name

</th>

<th>

Last name

</th>

<th>

Middle name

</th>

<th>

Position

</th>

<th>

Salary

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.Firstname)</td>

<td> @Html.DisplayFor(modelItem => item.LastName)</td>

<td> @Html.DisplayFor(modelItem => item.MiddleName)</td>

<td> @Html.DisplayFor(modelItem => item.Position)</td>

<td> @Html.DisplayFor(modelItem => item.Salary)</td>

@if (SignInManager.IsSignedIn(User) && User.IsInRole("admin"))

{

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id">Edit</a> |

<a **asp-action**="Delete" **asp-route-id**="@item.Id">Delete</a>

</td>

}

</tr>

}

</tbody>

</table>

<p>

<a **asp-action**="Create">Create</a>

</p>

<center>

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</center>

**Orders/Completed.cshtml**

@model IEnumerable<BusinessLayer.Models.OrderDTO>

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

@using Microsoft.AspNetCore.Identity

<h4>Order</h4>

<table class="table">

<thead>

<tr>

<th>

Date

</th>

<th>

Service

</th>

<th>

Price

</th>

<th>

Car id

</th>

<th>

Owner id

</th>

<th>

Inspector id

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

@if (item.IsCompleted == true)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.Date)</td>

<td> @Html.DisplayFor(modelItem => item.ServiceName)</td>

<td> @Html.DisplayFor(modelItem => item.Price)</td>

<td> @Html.DisplayFor(modelItem => item.CarId)</td>

<td> @Html.DisplayFor(modelItem => item.OwnerId)</td>

<td> @Html.DisplayFor(modelItem => item.InspectorId)</td>

</tr>

}

}

</tbody>

</table>

<p>

<a **asp-action**="Index">Back</a>

</p>

<center>

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</center>

**Orders/Create.cshtml**

@model BusinessLayer.Models.OrderDTO

<h4>Order</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Create">

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

<div class="form-group">

<label **asp-for**="Date" class="control-label">Date</label>

<input **asp-for**="Date" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="ServiceName" class="control-label">Service</label>

<input **asp-for**="ServiceName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Price" class="control-label">Price</label>

<input **asp-for**="Price" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="CarId" class="control-label">Car id</label>

<input **asp-for**="CarId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="OwnerId" class="control-label">Owner id</label>

<input **asp-for**="OwnerId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="InspectorId" class="control-label">Inspector id</label>

<input **asp-for**="InspectorId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="IsCompleted" class="control-label">Completed</label>

<input **asp-for**="IsCompleted" class="form-control" />

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

</div>

<div class="form-group">

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

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Orders /Delete.cshtml**

@model BusinessLayer.Models.OrderDTO

<h3>Are you sure?</h3>

<div>

<hr />

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="float:right; opacity:0.5" />

<dl class="row">

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Model.Date

</dd>

<dt class="col-sm-2">

<label **asp-for**="ServiceName">Service</label>

</dt>

<dd class="col-sm-10">

@Model.ServiceName

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Model.Price

</dd>

<dt class="col-sm-2">

<label **asp-for**="CarId">Car id</label>

</dt>

<dd class="col-sm-10">

@Model.CarId

</dd>

<dt class="col-sm-2">

<label **asp-for**="OwnerId">Owner id</label>

</dt>

<dd class="col-sm-10">

@Model.OwnerId

</dd>

<dt class="col-sm-2">

<label **asp-for**="InspectorId">Inspector id</label>

</dt>

<dd class="col-sm-10">

@Model.InspectorId

</dd>

<dt class="col-sm-2">

<label **asp-for**="IsCompleted">Completed</label>

</dt>

<dd class="col-sm-10">

@Model.IsCompleted

</dd>

</dl>

<form **asp-action**="DeleteById">

<input **type**="hidden" **asp-for**="Id" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a **asp-action**="Index">Back</a>

</form>

</div>

**Orders / Edit.cshtml**

@model BusinessLayer.Models.OrderDTO

<h4>Order</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Edit">

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

<input **type**="hidden" **asp-for**="Id" />

<div class="form-group">

<label **asp-for**="Date" class="control-label">Date</label>

<input **asp-for**="Date" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="ServiceName" class="control-label">Service</label>

<input **asp-for**="ServiceName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="Price" class="control-label">Price</label>

<input **asp-for**="Price" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="CarId" class="control-label">Car id</label>

<input **asp-for**="CarId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="OwnerId" class="control-label">Owner id</label>

<input **asp-for**="OwnerId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="InspectorId" class="control-label">Inspector id</label>

<input **asp-for**="InspectorId" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="IsCompleted" class="control-label">Completed</label>

<input **asp-for**="IsCompleted" class="form-control" />

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

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" /> |

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Orders / Index.cshtml**

@model IEnumerable<BusinessLayer.Models.OrderDTO>

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

@using Microsoft.AspNetCore.Identity

<table class="table">

<thead>

<tr>

<th>

Date

</th>

<th>

Service

</th>

<th>

Price

</th>

<th>

Car id

</th>

<th>

Owner id

</th>

<th>

Inspector id

</th>

<th>

Completed

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.Date)</td>

<td> @Html.DisplayFor(modelItem => item.ServiceName)</td>

<td> @Html.DisplayFor(modelItem => item.Price)</td>

<td> @Html.DisplayFor(modelItem => item.CarId)</td>

<td> @Html.DisplayFor(modelItem => item.OwnerId)</td>

<td> @Html.DisplayFor(modelItem => item.InspectorId)</td>

<td> @Html.DisplayFor(modelItem => item.IsCompleted)</td>

@if (SignInManager.IsSignedIn(User) && User.IsInRole("admin"))

{

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id">Edit</a> |

<a **asp-action**="Delete" **asp-route-id**="@item.Id">Delete</a>

</td>

}

</tr>

}

</tbody>

</table>

<p>

<a **asp-action**="Create">Create</a> | <a **asp-action**="Completed">Show completed</a>

</p>

<center>

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</center>

**Owners/Create.cshtml**

@model BusinessLayer.Models.OwnerDTO

<h4>Owner</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Create">

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

<div class="form-group">

<label **asp-for**="Firstname" class="control-label">First name</label>

<input **asp-for**="Firstname" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label">Last name</label>

<input **asp-for**="LastName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="MiddleName" class="control-label">Middle name</label>

<input **asp-for**="MiddleName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="PhoneNum" class="control-label">Phone number</label>

<input **asp-for**="PhoneNum" class="form-control" />

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

</div>

<div class="form-group">

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

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Owners /Delete.cshtml**

@model BusinessLayer.Models.OwnerDTO

<h3>Are you sure?</h3>

<div>

<hr />

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="float:right; opacity:0.5" />

<dl class="row">

<dt class="col-sm-2">

<label **asp-for**="Firstname">First name</label>

</dt>

<dd class="col-sm-10">

@Model.Firstname

</dd>

<dt class="col-sm-2">

<label **asp-for**="LastName">Last name</label>

</dt>

<dd class="col-sm-10">

@Model.LastName

</dd>

<dt class="col-sm-2">

<label **asp-for**="MiddleName">Middle name</label>

</dt>

<dd class="col-sm-10">

@Model.MiddleName

</dd>

<dt class="col-sm-2">

<label **asp-for**="PhoneNum">Phone number</label>

</dt>

<dd class="col-sm-10">

@Model.PhoneNum

</dd>

</dl>

<form **asp-action**="DeleteById">

<input **type**="hidden" **asp-for**="Id" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a **asp-action**="Index">Back</a>

</form>

</div>

**Owners / Edit.cshtml**

@model BusinessLayer.Models.OwnerDTO

<h4>Owner</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Edit">

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

<input **type**="hidden" **asp-for**="Id" />

<div class="form-group">

<label **asp-for**="Firstname" class="control-label">First name</label>

<input **asp-for**="Firstname" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label">Last name</label>

<input **asp-for**="LastName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="MiddleName" class="control-label">Middle name</label>

<input **asp-for**="MiddleName" class="form-control" />

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

</div>

<div class="form-group">

<label **asp-for**="PhoneNum" class="control-label">Phone number</label>

<input **asp-for**="PhoneNum" class="form-control" />

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

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" /> |

<a **asp-action**="Index">Back</a>

</div>

</form>

</div>

<div class="col-md-4">

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Owners / Index.cshtml**

@model IEnumerable<BusinessLayer.Models.OwnerDTO>

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

@using Microsoft.AspNetCore.Identity

<table class="table">

<thead>

<tr>

<th>

First name

</th>

<th>

Last name

</th>

<th>

Middle name

</th>

<th>

Phone number

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td> @Html.DisplayFor(modelItem => item.Firstname)</td>

<td> @Html.DisplayFor(modelItem => item.LastName)</td>

<td> @Html.DisplayFor(modelItem => item.MiddleName)</td>

<td> @Html.DisplayFor(modelItem => item.PhoneNum)</td>

@if (SignInManager.IsSignedIn(User) && User.IsInRole("admin"))

{

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id">Edit</a> |

<a **asp-action**="Delete" **asp-route-id**="@item.Id">Delete</a>

</td>

}

</tr>

}

</tbody>

</table>

<p>

<a **asp-action**="Create">Create</a>

</p>

<center>

<img class="" src="https://i.pinimg.com/564x/b7/b8/58/b7b85879316062886c12f5a368b0a085.jpg" style="opacity:0.5" />

</center>

**Shared/\_Layout.cshtml**

@using Microsoft.AspNetCore.Identity

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Service station</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Service station </a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<**partial** **name**="\_LoginPartial" />

<ul class="navbar-nav flex-grow-1">

@if (SignInManager.IsSignedIn(User))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Cars" **asp-action**="Index">Car</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Owners" **asp-action**="Index">Owner</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Inspectors" **asp-action**="Index">Inspector</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Orders" **asp-action**="Index">Order</a>

</li>

}

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2020 - Service station

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script **src**="~/js/site.js" **asp-append-version**="true"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**Shared /\_LoginPartial.cshtml**

@using Microsoft.AspNetCore.Identity

@inject SignInManager<DataLayer.Models.Employee> SignInManager

@inject UserManager<DataLayer.Models.Employee> UserManager

<ul class="navbar-nav">

@if (SignInManager.IsSignedIn(User))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Manage/Index" title="Manage">@User.Identity.Name</a>

</li>

<li class="nav-item">

<form class="form-inline" **asp-area**="Identity" **asp-page**="/Account/Logout" **asp-route-returnUrl**="@Url.Action("Index", "Home", new { area = "" })">

<button type="submit" class="nav-link btn btn-link text-dark">Logout</button>

</form>

</li>

}

else

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Register">Register</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="Identity" **asp-page**="/Account/Login">Login</a>

</li>

}

</ul>

**appsettings.json**

{

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=ServiceStationWebAppPractice;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

**IdentityInitializer.cs**

using DataLayer.Models;

using Microsoft.AspNetCore.Identity;

using System.Threading.Tasks;

namespace ServiceStationWebApp

{

public class IdentityInitializer

{

private const string \_email = "mashaprotskaya@gmail.com";

private const string \_password = "123456Aa\_";

private const string \_surname = "Protskaya";

private const string \_firstName = "Masha";

public static async Task InitializeAsync(UserManager<Employee> userManager, RoleManager<IdentityRole> roleManager)

{

if (await roleManager.FindByNameAsync(Roles.Admin) == null)

{

await roleManager.CreateAsync(new IdentityRole(Roles.Admin));

}

if (await roleManager.FindByNameAsync(Roles.User) == null)

{

await roleManager.CreateAsync(new IdentityRole(Roles.User));

}

if (await userManager.FindByNameAsync(\_email) == null)

{

var admin = new Employee { Surname = \_surname, FirstName = \_firstName, Email = \_email, UserName = \_email };

IdentityResult result = await userManager.CreateAsync(admin, \_password);

if (result.Succeeded)

{

await userManager.AddToRoleAsync(admin, Roles.Admin);

}

}

}

}

}

**Program.cs**

using DataLayer.Models;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Identity;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using NLog.Web;

using System;

namespace ServiceStationWebApp

{

public class Program

{

public static async System.Threading.Tasks.Task Main(string[] args)

{

var logger = NLogBuilder.ConfigureNLog("nlog.config").GetCurrentClassLogger();

try

{

logger.Debug("init main");

var host = CreateHostBuilder(args).Build();

using (var scope = host.Services.CreateScope())

{

var services = scope.ServiceProvider;

try

{

var userManager = services.GetRequiredService<UserManager<Employee>>();

var rolesManager = services.GetRequiredService<RoleManager<IdentityRole>>();

await IdentityInitializer.InitializeAsync(userManager, rolesManager);

}

catch (Exception ex)

{

var \_logger = services.GetRequiredService<ILogger<Program>>();

\_logger.LogError(ex, "An error occurred while seeding the database.");

}

}

host.Run();

}

catch (Exception exeption)

{

logger.Error(exeption, "Stopped program because of exception");

throw;

}

finally

{

NLog.LogManager.Shutdown();

}

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

})

.ConfigureLogging(logging =>

{

logging.ClearProviders();

logging.SetMinimumLevel(LogLevel.Trace);

})

.UseNLog();

}

}

**Roles.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ServiceStationWebApp

{

public class Roles

{

public const string Admin = "admin";

public const string User = "user";

}

}

**Startup.cs**

using AutoMapper;

using BusinessLayer;

using DataLayer;

using DataLayer.Models;

using DataLayer.Repository;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using ServiceStationWebApp.Data;

using System.Reflection;

namespace ServiceStationWebApp

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ServiceStationContext>(options =>

options.UseSqlServer(

Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<Employee, IdentityRole>()

.AddEntityFrameworkStores<ServiceStationContext>();

services.AddControllersWithViews();

services.AddRazorPages();

var bl = Assembly.Load("BusinessLayer");

services

.AddTransient(typeof(IRepository<>), typeof(GenericRepository<>))

.Scan(scan => scan

.FromAssemblies(bl)

.AddClasses(classes => classes.Where(type => type.Name.EndsWith("Service")))

.AsSelf()

.WithTransientLifetime())

.AddAutoMapper(typeof(ServiceStationProfile));

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseDatabaseErrorPage();

}

else

{

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

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

endpoints.MapRazorPages();

});

}

}

}