

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет компьютерных технологий

Кафедра программного обеспечения информационных технологий

К защите допустить:

Заведующий кафедрой ПОИТ

_____ Н.В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

**ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА МОБИЛЬНЫХ УСТРОЙСТВ ЗАО
«КЬЮЛИКС СИСТЕМС» НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C#**

БГУИР ДП 1-40 01 01 01 095 ПЗ

Студент	А.В. Третьякова
Руководитель	Д.В. Горбачев
Консультанты: <i>от кафедры</i> <i>по экономической части</i>	Д.В. Горбачев С.В. Наркевич
Нормоконтролер	С.В. Болтак
Рецензент	

Минск 2022

РЕФЕРАТ

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА МОБИЛЬНЫХ УСТРОЙСТВ ЗАО «КЬЮЛИКС СИСТЕМС» НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С#: дипломный проект/ А.В. Третьякова. – Минск: БГУИР, 2022, – п.з. – 145 с., чертежей (плакатов) – 6 л. формата А1.

Объектом исследования является процесс учета мобильных устройств ЗАО «Кьюликс Системс».

Цель дипломного проекта: проектирование и разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

Разработка данного программного средства обеспечит возможность вести учет мобильных устройств ЗАО «Кьюликс Системс», что упростит организацию работы сотрудников организации. Использование веб-приложения сократит трудозатраты пользователей за счет снижения трудоемкости выполнения «ручных» операций и сокращения времени на устранение инцидентов (сбой в работе устройства, нежелательное обновление операционной системы).

Проанализированы существующие программные средства для учета компьютерной техники в организации, выполнена постановка задач.

Выполнено моделирование предметной области, разработана информационная модель базы данных, диаграмма вариантов использования и функциональные требования к программному средству. В процессе проектирования программного средства разработана физическая модель базы данных, а также ряд диаграмм в нотации UML. На основании проектной документации выполнена реализация программного средства. Проведено тестирование и разработана методика использования программного средства. Уделено внимание вопросам технико-экономического обоснования программного средства.

В разделе технико-экономического обоснования были произведены расчеты затрат, связанных с реализацией проекта, а также рентабельности разработки проекта. Проведенные расчеты показали экономическую целесообразность проекта.

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И
РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий

Факультет КТ Кафедра ПОИТ
Специальность 1–40 01 01 Специализация 01

УТВЕРЖДАЮ

« » Н.В. Лапицкая
« » 20 г.

ЗАДАНИЕ
по дипломному проекту студента

Третьяковой Анастасии Владимировны
(фамилия, имя, отчество)

1. Тема проекта: **ВЕБ–ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА МОБИЛЬНЫХ
УСТРОЙСТВ ЗАО «КЬЮЛИКС СИСТЕМС» НА ЯЗЫКЕ
ПРОГРАММИРОВАНИЯ C#**

утверждена приказом по университету от «01» ноября 2021 г. № 224-и

2. Срок сдачи студентом законченной работы 03 января 2022 г.

3. Исходные данные к проекту Тип операционной системы: Windows 11;
Языки программирования: C#;
Назначение разработки: учет мобильных устройств ЗАО «Кьюликс Системс».

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов)
Введение

1. Анализ литературных источников и формирование требований к проектируемому программному средству

2. Моделирование программного средства

3. Проектирование программного средства

4. Тестирование программного средства

5. Методика использования программного средства

6. Техничко-экономическое обоснование разработки и использования программного средства

Заключение

Список использованных источников

Приложение А. Текст программы

5. Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала)

Алгоритм проверки уровня доступа на стороне сервера. Схема алгоритма – формат А1, лист 1.

Алгоритм добавления сотрудника. Схема алгоритма – формат А1, лист 1.

Алгоритм добавления устройства. Схема алгоритма – формат А1, лист 1.

Диаграмма вариантов использования. Плакат – формат А1, лист 1.

Схема базы данных. Плакат – формат А1, лист 1.

Диаграмма классов. Плакат – формат А1, лист 1.

6. Техничко-экономическое обоснование дипломного проекта:

Описание функций, назначения и потенциальных пользователей программного средства

Расчет затрат на разработку программного средства

Оценка экономического эффекта от разработки и применения программного средства для собственных нужд организации

Задание выдал _____ / С.В. Наркевич /

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов дипломного проекта (работы)	Объём этапа в %	Срок выполнения этапа	Примечание
Анализ предметной области, разработка технического задания	10	27.10 – 09.11	
Моделирование предметной области	20	10.11 – 18.11	
Проектирование программного средства	20	19.11 – 25.11	
Тестирование программного средства	10	26.11 – 02.12	
Методика использования разработанного программного средства	20	02.12 – 16.12	
Оформление пояснительной записки и графического материала	20	17.12 – 31.12	

Дата выдачи задания 01 ноября 2021 г. Руководитель _____ / Д.В. Горбачев /

Задание принял к исполнению _____ / А.В. Третьякова /

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ ...	8
1.1 Описание предметной области	8
1.2 Анализ существующих аналогов.....	10
1.3 Постановка цели и задач	16
1.4 Входные данные	177
1.5 Выходные данные	17
2 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА	18
2.1 Функциональные требования.....	18
2.2 Описание функциональности программного средства	18
2.3 Разработка информационной модели	20
2.4 Разработка модели взаимодействия пользователя с интерфейсом.....	22
2.5 Разработка технических требований к программному средству	23
3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА.....	24
3.1 Обоснование выбора среды разработки	24
3.2 Разработка структурной схемы программного средства	27
3.3 Разработка структуры классов.....	28
3.4 Разработка физической модели данных	30
3.5 Проектирование алгоритмов работы программного средства	34
4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА.....	37
5 МЕТОДИКА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА	44
5.1 Авторизация пользователя	44
5.2 Детальная информация об устройстве и его статус	46
5.3 Администратор	47
6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА	55
6.1 Описание функций, назначения и потенциальных пользователей программного средства.....	55
6.2 Расчет затрат на разработку программного средства.....	55
6.3 Оценка экономического эффекта от разработки и применения программного средства для собственных нужд организации	58
ЗАКЛЮЧЕНИЕ	59
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	60
ПРИЛОЖЕНИЕ А (Обязательное) Текст программы.....	6161

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Закрытое акционерное общество (ЗАО) – акционерное общество, акции которого распределяются только среди учредителей или заранее определенного круга лиц (в противоположность открытому) [13].

Авторизация – предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

Аутентификация – проверка подлинности предъявленного пользователем идентификатора.

БД – база данных.

ОС – операционная система.

Программа – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.

Программное средство (ПС) – совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

API – application programming interface (интерфейс программирования приложений).

JWT – JSON Web Token (это JSON объект, который определен в открытом стандарте RFC 7519).

LINQ – Language-Integrated Query (Интегрированный язык запросов).

Веб-приложение – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера, которое выполняет функцию контроля пользования мобильными устройствами [3].

Мобильное устройство – переносное устройство, принадлежащее организации, с операционной системой IOS или Android.

Администратор – сотрудник ЗАО «Кьюликс Системс», который ведет учет мобильных устройств.

Держатель (сотрудник) – пользователь мобильного устройства в текущий момент времени.

Статус – статус мобильного устройства в веб-приложении.

ВВЕДЕНИЕ

Актуальность учета компьютерной и другой техники в организации возрастает при наличии большого количества компьютеров, офисной техники и другого оборудования.

Автоматизация сфер деятельности организации, будь то сбор данных или управление различными технологическими процессами, позволяет ускорить работу, сделать ее более точной и эффективной, избежать потерь нужной информации, ошибок персонала, дублирования документов. Для эффективности работы по учету техники необходимо проводить мероприятия по обновлению материально-технической и информационной базы, внедрению современных методов управления с использованием автоматизированных систем, созданию базы данных о технике организации, его своевременному пополнению, оперативному представлению необходимой информации пользователям.

Целью дипломного проекта является разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

Задачи дипломного проекта:

- изучить аналоги программного средства по учету компьютерной либо любой другой техники в организации;
- спроектировать информационную структуру разрабатываемого программного средства;
- разработать схему базы данных и пользовательский интерфейс для доступа к данным;
- провести расчет технико-экономических показателей разрабатываемого программного средства.

Объектом дипломного проекта является процесс учета мобильных устройств ЗАО «Кьюликс Системс».

Предметом дипломного проекта является автоматизация учета мобильных устройств ЗАО «Кьюликс Системс».

Практическая значимость дипломного проекта состоит в том, что использование веб-приложения сократит трудозатраты пользователей за счет снижения трудоемкости выполнения «ручных» операций и сокращения времени на устранение инцидентов (сбой в работе устройства, нежелательное обновление операционной системы).

1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1 Описание предметной области

Каждая организация приходит к тому, что требуется навести полный порядок в инфраструктуре организации, получить контроль над перемещениями техники, спланировать сервисные работы, заказы и закупки, списание техники. Необходимо иметь возможность в любой момент централизованно получать полную информацию о состоянии подотчетного оборудования.

Учет – это количественное отражение событий и фактов, используемое для управления обществом на различных организационных уровнях. Учет представляет собой сбор, обработку, классификацию, систематизацию и отражение информации в специальных регистрах на каких-либо носителях. В экономике используются различные виды учета: бухгалтерский, оперативно-технический, статистический и другие. Общая цель всех видов учета – упорядочение большого количества информации для эффективного использования в управленческих решениях, формирования отчетности и сохранение информации для архива [6].

Оперативная информация используется для принятия управленческих решений, составления и анализа финансовой отчетности организации, долгосрочного и текущего планирования, исполнения налоговых обязательств. Отличительной особенностью оперативной информации является ее конфиденциальность. Оперативная информация должна быть доступна только для сотрудников организации. Управленческий (оперативный) учет – это внутреннее дело каждой организации. Для ведения этого учета могут быть использованы любые формы и регистры, разработанные самой организацией [7].

Все виды учета являются составной частью интегрированной управленческой информационной системы. При создании такой системы необходимо ответить на следующие вопросы:

- какая информация нужна (информация различается по степени детализации и стилю представления. Степень детализации оперативной информации зависит от иерархии управления. Детализация и стиль представления оперативной информации зависят от получателя. Высшему управленческому персоналу требуется обобщенная оперативная информация. Она должна быть наиболее интегрированной и с более длительным периодом представления. Обобщенная оперативная информация может представляться ежемесячно);

- кому нужна эта информация (оперативная информация предназначена для внутреннего пользования. Состав пользователей

информации определяется структурой организации, которая зависит от вида деятельности. Пользователей информации можно разделить на две составляющие: управленческий персонал и непосредственно исполнители);

- почему нужна эта информация (оперативная информация необходима для принятия оперативных управленческих решений управленческим персоналом, реализации управленческих решений специалистами организации, определения отклонений достигнутых результатов от прогнозируемых, разработки мероприятий по выявленным отклонениям);

- когда она необходима (информация должна быть достоверной и достаточной для принятия управленческих решений);

- где получить информацию (информацию различают как первичную и вторичную. Источник информации, определяемый схемой документооборота, утвержденной руководством организации, можно отнести к вторичной информации. Внешними источниками информации могут быть поставщики и покупатели, конкуренты, реклама, государственные органы и др. Таким образом, наблюдение за внешней средой позволяет собрать первичную информацию, необходимую для бизнеса) [7].

Проектирование программного средства должно предусматривать использование информационных технологий и способы обмена информацией.

Требования к качеству информации. Основными принципами учета оперативной информации являются качественные характеристики этой информации и принципы ее учета. Основное требование к учетной информации – это ее полезность для принятия решений различными группами пользователей. Полезность информации определяется специальными характеристиками:

- ясность;
- уместность;
- содержательность;
- существенность;
- своевременность;
- полнота отражения;
- логичность;
- достоверность [7].

Таким образом, качественные характеристики оперативной учетной информации предопределяют полезность оперативной отчетности для пользователей. В нахождении оптимального сочетания всех характеристик проявляется профессионализм специалистов по контролю и экономистов-аналитиков. Соблюдение всех вышеперечисленных требований к качеству информации должно сочетаться с ограничением затрат и выгоды, которое состоит в том, что выгода от информации должна быть больше, чем затраты на ее получение. Выполнение требований к качеству оперативной учетной

информации должно дать в результате правдивую, полную, полезную оперативную отчетность о деятельности организации.

1.2 Анализ существующих аналогов

Сейчас создано много различных программных средств для учета различной техники в организации: с простым и сложным интерфейсом, бесплатные и платные программные средства. Но каждая программа содержит различные функции и распространяется по-разному.

1.2.1 Программное средство «Hardware Inspector»

ПС «Hardware Inspector» предназначено для инвентарного учета компьютеров и другой техники в организациях. Ориентировано на выполнение работ, связанных с учетом и планированием компьютерного (и иного) аппаратного обеспечения. Оно позволяет всегда быть в курсе всей информации о компьютерном парке компании, получать разнообразные отчеты, планировать его обслуживание, ремонт и обновление [8].

Программное средство «Hardware Inspector» решает задачи автоматизации инвентарного учета компьютерной техники и комплектующих, с возможностью хранения всей истории перемещений и обслуживания. Механизм ревизий рабочих мест предохраняет компьютеры и комплектующие от хищения и подмены. Детальный контроль за параметрами конфигурации компьютера обеспечивает свободу и оперативность действий по планированию модернизации и перераспределения устройств [8].

Функционал «Hardware Inspector» не ограничивается одним лишь учетом компьютеров – система позволяет автоматизировать почти все аспекты деятельности ИТ-подразделения. Набор из более чем 30 встроенных отчетов позволит оценить эффективность его работы и предоставить нужные отчеты вышестоящему руководству.

Основные возможности программного средства «Hardware Inspector»:

- учет комплектующих;
- отслеживание истории перемещения устройств, их ремонта, профилактики и инвентаризации;
- возможность не только ручного ввода данных, но и импорта информации из отчетов программ анализа конфигурации компьютеров AIDA32, EVEREST, ASTRA И ASTRA32;
- учет лицензий на программное средство;
- мониторинг инсталляции пользователями запрещенного ПС;
- мониторинг изменений, происходящих в конфигурациях компьютеров (аудит рабочих мест);
- учет заявок от пользователей;
- инвентаризация устройств с использованием штрих-кодов;

- гибкое разграничение доступа к данным;
- мощные механизмы поиска устройств, лицензий и прочего;
- большой набор настраиваемых отчетов, экспорт в различные форматы;

- поддержка многопользовательской работы с базой данных в сети [8].

Уникальность ПС «Hardware Inspector» заключается в возможности вести учет не просто текущего состояния параметров компьютера, а всей истории жизни каждого устройства.

Механизм поиска поможет найти интересующее устройство (группу устройств) по заданному критерию. В фильтре поиска могут участвовать такие параметры устройств, как тип, модель, производитель, продавец, инвентарный номер, серийный номер, произвольная строка поиска и так далее (смотрите рисунок 1.1).

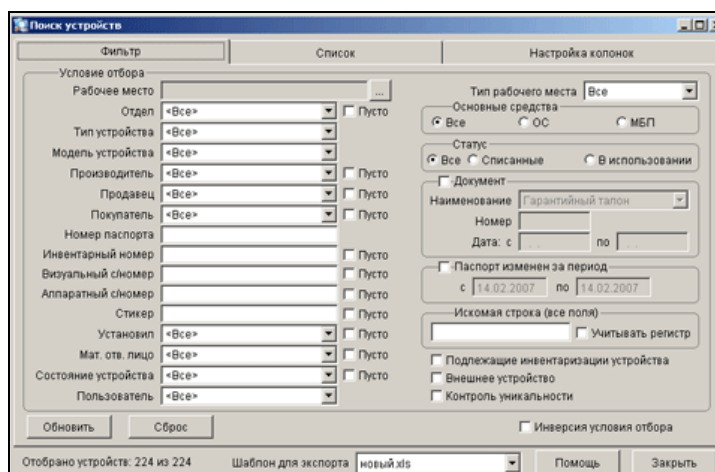


Рисунок 1.1 – Поиск устройства в «Hardware Inspector»

Достоинствами данной системы являются полный учет комплектующих, а не просто описание параметров рабочих станций; сопровождение системы после покупки и регулярное обновление системы (смотрите справочник типов и моделей устройств на рисунке 1.2).

Имеет гибкую систему разграничения прав доступа, большой набор встроенных отчетов, а также механизм создания отчетов, требуемых компании.

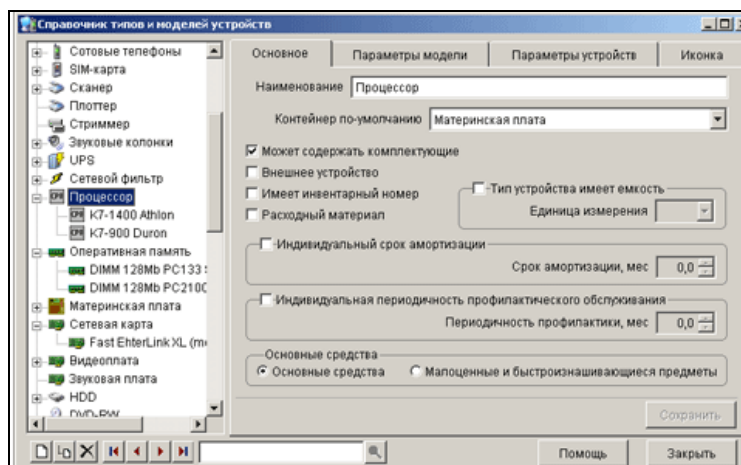


Рисунок 1.2 – Справочник типов и моделей устройств в «Hardware Inspector»

Данная система имеет закрытую архитектуру, в связи с чем отсутствует возможность вносить изменения в функционал системы. Такой недостаток не позволяет в полной мере внедрить систему в организацию, так же функционал, имеющийся в «Hardware Inspector», излишен для нужд ЗАО «Кьюликс Системс».

1.2.2 Программное средство «Инвентаризация сети и учета и компьютеров»

Программное средство «Инвентаризации сети и учет компьютеров» является разработкой компании Clear Apps.

Основная задача, которую выполняет система – это инвентаризация компьютерных сетей. В инвентаризацию сети входят учёт установленного программного средства, а также регистрация всего аппаратного парка сети [9].

Данная система позволяет вести учёт всей компьютерной техники и программного средства сети. Эта процедура просто необходима при планировании обновления аппаратного обеспечения, добавлении новых рабочих мест или закупки дополнительных аппаратных или программных ресурсов (смотрите главную страницу ПС на рисунке 1.3) [9].

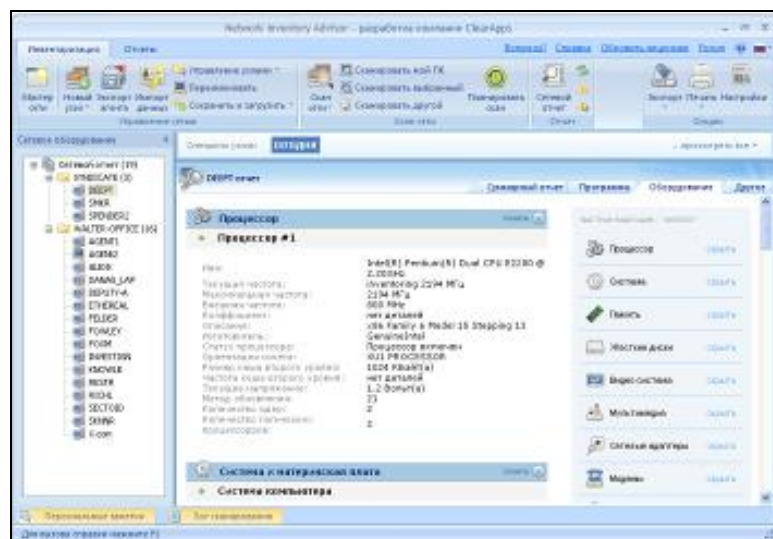


Рисунок 1.3 – Главная страница ПС «Инвентаризации сети и учет компьютеров»

Система также предназначена для контроля использования аппаратных средств и лицензий ПС. Инвентаризация сети необходима для контроля соответствия установленного программного и аппаратного средства с заданным стандартным набором для каждого рабочего места.

Данное программное средство имеет систему отчётности выполнения плановых мероприятий по всей ИТ-инфраструктуре компьютерной сети.

Имеется возможность добавлять в отчёты по инвентаризации компьютеры, которые не подключены к сети. Для этого в системе предусмотрен специальный модуль, обрабатывающий все данные, которые можно будет импортировать в интерфейс программы.

Для данной системы можно выделить следующие достоинства:

- автоматизация – сканирование сети осуществляется автоматически с высокой скоростью;
- наглядность – группировка собранной информации производится как для всей сети в целом, так и для каждого компьютера в отдельности;
- гибкость – имеется возможность добавлять специальные поля для каждого компьютера;
- удобство – собранные данные можно распечатать или экспортировать в нужный формат;
- отчёты – генератор отчётов и предустановленные шаблоны максимально ускорят процедуру построения отчётов по сети [9].

Недостатком системы можно выделить сложность ее внедрения: для внесения данных о компьютере, на него необходимо установить специальный модуль, сканирующий компьютер. Лишь после этого компьютер будет внесен в базу данных.

Кроме того, данный модуль проработан недостаточно хорошо, вследствие чего некоторые данные об оборудовании приходится вносить вручную. Для нужд ЗАО «Кьюликс Системс» данная система является

слишком громоздкой и трудозатратой в плане переработки для собственных нужд.

1.2.3 Программное средство «Учет компьютеров»

Программное средство предназначено для учета компьютеров, оргтехники и других объектов в организации, разработана компанией ООО «Простой софт».

ПС позволяет фиксировать производимые ремонты, замены расходных материалов и частей, профилактические работы, установленное программное средство. Все объекты учета можно закреплять за сотрудниками с разбиением на отделы и филиалы (рабочее окно ПС «Учет компьютеров» смотрите на рисунке 1.4) [9].

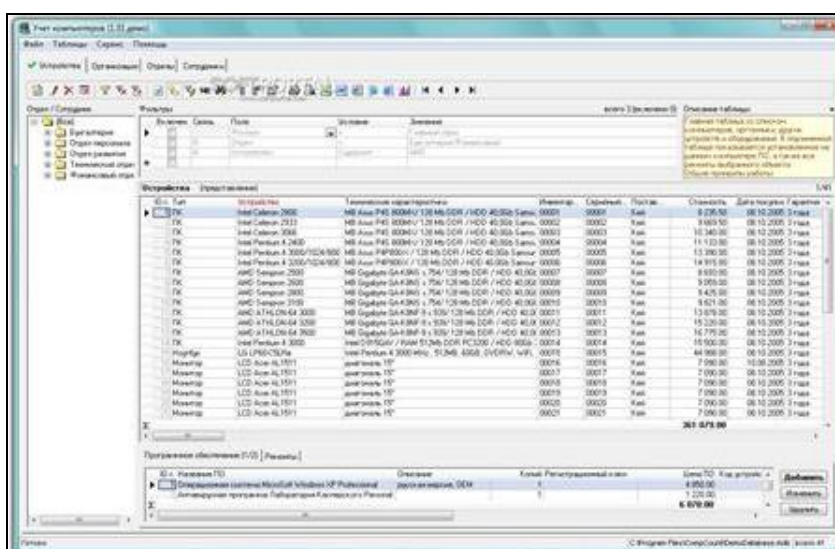


Рисунок 1.4 – Рабочее окно программы «Учет компьютеров»

ПС необходимо системным администраторам, руководителям ИТ – отделов, владельцам компьютерных магазинов, держателям домашних сетей, обладателям большого количества комплектующих и компьютерной техники.

Основные функции:

- ведение базы компьютеров и оргтехники: в базе данных содержится информация обо всех компьютерах и оргтехнике организации;
- предусмотрены удобные способы сортировки и фильтрации данных;
- поддерживается экспорт таблиц в MS Excel, MS Word или текстовый формат CSV, а также импорт из других источников данных;
- фиксация ремонтов и апгрейдов, а также контроль гарантийных сроков;
- учет установленного программного средства;
- закрепление объектов за ответственными сотрудниками: все объекты

учета можно закреплять за ответственными сотрудниками с разбиением на отделы и филиалы [9].

Данная система позволяет не только вести учет оборудования, но и составлять бюджет отдела: спланировать потребность в необходимом количестве расходных материалов на определённый период, комплектующих для резерва на случай возможных отказов и тому подобное.

Многопользовательский режим работы позволяет распределить зоны ответственности: кто-то будет заносить в базу информацию о новом поступившем оборудовании, кто-то о поставщиках и прочих деловых партнерах. В результате обеспечивается согласованная работа сотрудников различных подразделений. Права на просмотр и изменение информации в системе настраиваются администратором в зависимости от функциональных обязанностей каждого пользователя.

Преимуществами системы можно выделить следующие возможности и функции:

- возможность изменения структуры данных – добавления таблиц, полей, связей, что очень важно при внедрении системы на предприятие;
- возможность создания шаблонов отчетов (счетов, актов и других документов);
- используется генерация документов Word на основе шаблонов (счетов, актов и других документов);
- поддерживается многопользовательский режим;
- настраиваемый пользовательский интерфейс [9].

Но при этом, анализируя отзывы пользователей, было выявлено, что при большом компьютерном парке работа базы данных и системы в целом начинает работать медленно, возникают различные ошибки и прерывания в работе системы. Для нужд ЗАО «Кьюликс Системс» данная система является слишком громоздкой и трудозатратой в плане переработки для собственных нужд.

1.2.4 Вывод

Таким образом, проанализировав данные разработки, можно сделать следующие выводы.

Все системы используют автоматическое сканирование компьютеров для получения информации обо всех компьютерах сети. Для этого могут быть использованы как сторонние приложения (Everest, ASTRA и тому подобные), запускаемые на рабочих местах удаленно с помощью скриптов, так и встроенные модули системы.

Аналогичным образом происходит инвентаризация программного средства, установленного на компьютерах сети предприятия. Данная возможность также характерна для всех рассматриваемых систем и является неотъемлемой частью системы учета компьютеров.

Все системы поддерживают интеграцию с различными базами данных, а также возможность импорта отчетов во все распространенные форматы файлов. Кроме того, в базах данных рассмотренных систем предусмотрена возможность привязки каждого компьютера к конкретному сотруднику и отделу.

Обязательной характеристикой оборудования при инвентаризации является информация о гарантийном сроке, дате поступления и поставщике оборудования.

Также неотъемлемой частью всех систем учета является генерация разнообразных отчетов. При этом имеются не только встроенные, шаблонные, отчеты, но и у большинства систем имеется возможность создания собственных, индивидуальных, отчетов.

Анализируя выделенные у данных систем достоинства и недостатки, можно сказать, что нет необходимости в разработке или же в приобретении сторонних громоздких систем для учета мобильных устройств для нужд ЗАО «Кьюликс Системс», так как основная цель разрабатываемой системы – это учет местоположения девайса и сокращение времени реакции на какие-либо инциденты, связанные с девайсом, за счет того, что все отметки будут вестись в одном месте. Так же нет необходимости в предоставлении отчетности руководству о перемещениях мобильных устройств между сотрудниками.

1.3 Постановка цели и задач

Как показал анализ программных решений в области учета компьютерной техники, они обладают следующими недостатками:

- закрытая архитектура, в связи с чем отсутствует возможность вносить изменения в функционал системы;
- сложность внедрения и переработки имеющегося функционала ПС для нужд ЗАО «Кьюликс Системс»;
- громоздкость, то есть имеющийся функционал не нужен для учета мобильных устройств в ЗАО «Кьюликс Системс»;
- высокая стоимость.

Целью дипломного проекта является разработка веб-приложения для учета мобильных устройств ЗАО «Кьюликс Системс».

ПС должно обеспечивать выполнение следующих функций:

- авторизация сотрудника;
- добавление, редактирование и удаление информации о сотрудниках (доступно только администратору);
- добавление, редактирование и удаление информации о мобильных устройствах (доступно только администратору);
- изменение статуса мобильного устройства;
- просмотр информации о мобильных устройствах;

- просмотр информации о сотрудниках (доступно только администратору);
- комментирование состояния мобильного устройства (необходимо для быстрого информирования о состоянии мобильного устройства (сбой в работе, нежелательное обновление ОС)).

1.4 Входные данные

К входной информации будут относиться все вводимые сотрудником и администратором данные.

Администратор при добавлении и редактировании информации о сотрудниках и мобильных устройствах будет указывать о них соответствующую информацию, также при авторизации сотрудник будет указывать свою персональную информацию (логин и пароль).

Так же сотрудник может оставлять комментарии по поводу состояния мобильного устройства.

Входной информацией будут является:

- данные для добавления, редактирования и удаления информации о сотрудниках;
- данные для добавления, редактирования и удаления информации о мобильных устройствах;
- данные для авторизации в системе;
- комментарии по статусу мобильного устройства.

1.5 Выходные данные

В качестве выходной информации будут выступать данные после добавления и сохранения отредактированной информации о сотрудниках и мобильных устройствах.

Система будет отдавать в качестве выходной информации список сотрудников (доступно только для администратора) и список мобильных устройств.

При изменении статуса мобильного устройства будет назначаться соответствующее мобильное устройство на сотрудника, который его взял.

К выходной информации также относятся системные данные: сессии и токены авторизации, ошибки системы.

К выходной информации относится:

- статус мобильного устройства;
- сессия и JWT;
- данные о сотрудниках;
- данные о мобильных устройствах;
- ошибки.

2 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Функциональные требования

Функциональным назначением разрабатываемого веб-приложения является предоставление возможности вести учет информации о мобильных устройствах ЗАО «Кьюликс Системс».

В качестве пользователя программного средства может выступать сотрудник ЗАО «Кьюликс системс», который добавлен администратором в данную систему и имеет доступ к персональному компьютеру с доступом к сети интернет. Для использования программного средства не требуется специальная подготовка или обучение пользователей.

Предполагается возможность одновременной эксплуатации разрабатываемого решения широким кругом пользователей.

Исходя из предполагаемого использования, можно заключить, что проектируемое программное средство должно реализовывать следующие группы функций:

- авторизация сотрудника;
- добавление, редактирование и удаление информации о сотрудниках (доступно только администратору);
- добавление, редактирование и удаление информации о мобильных устройствах (доступно только администратору);
- изменение статуса мобильного устройства;
- просмотр информации о мобильных устройствах;
- просмотр информации о сотрудниках (доступно только администратору);
- комментирование состояния мобильного устройства (необходимо для быстрого информирования о состоянии мобильного устройства (сбой в работе, нежелательное обновление ОС)).

2.2 Описание функциональности программного средства

Диаграмма вариантов использования – это наиболее общее представление функционального назначения системы.

Диаграмма вариантов использования призвана ответить на главный вопрос моделирования: «что делает система во внешнем мире?». На диаграмме применяются два типа основных сущностей: варианты использования и действующие лица, между которыми устанавливаются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования;
- обобщение между действующими лицами;
- обобщение между вариантами использования;

– зависимости (различных типов) между вариантами использования [1].

В системе можно выделить следующие действующие лица:

- держатель (сотрудник);
- администратор.

На рисунке 2.1 представлена общая диаграмма вариантов использования.

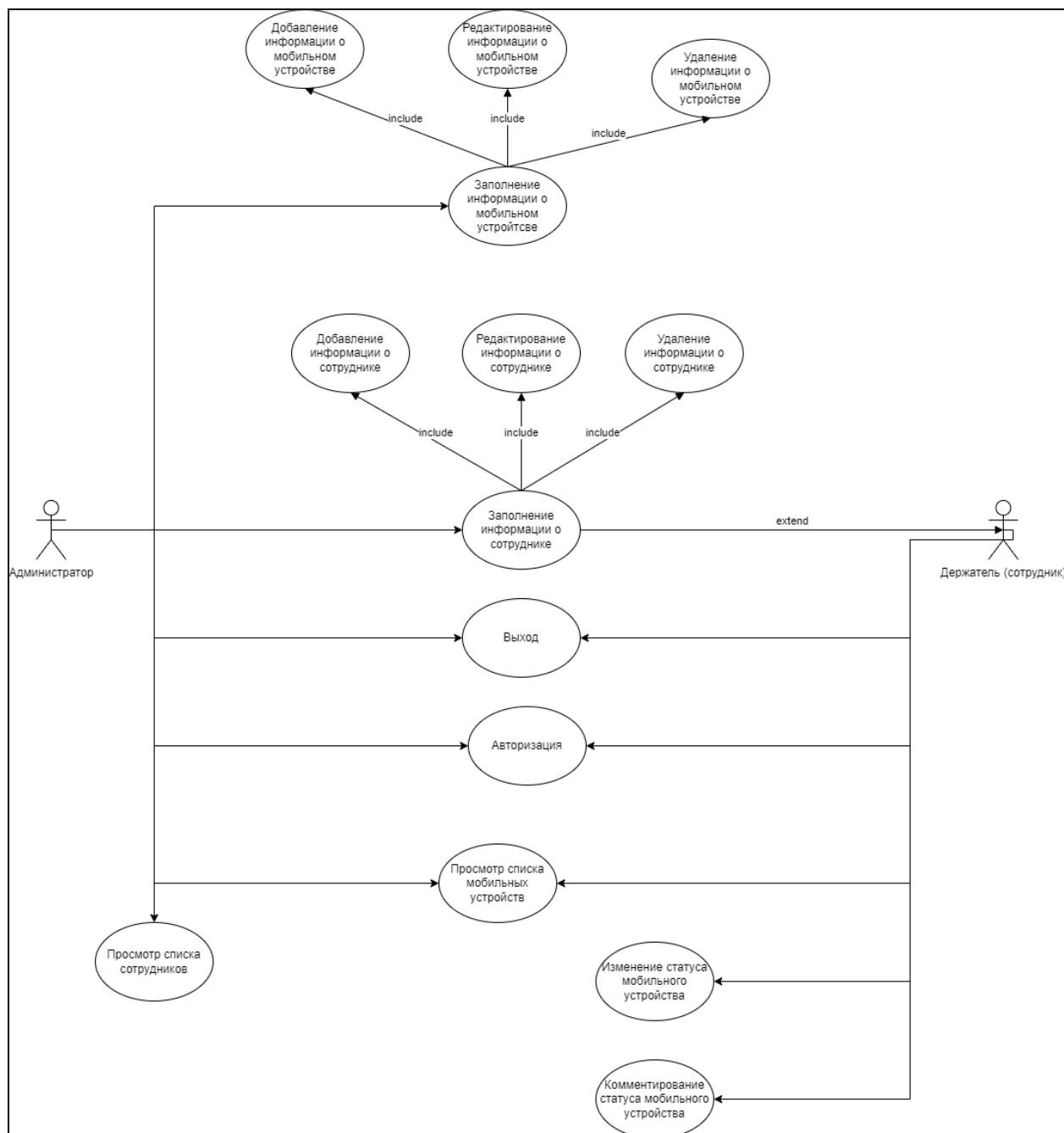


Рисунок 2.1 – Диаграмма вариантов использования

Учет мобильных устройств ЗАО «Кьюликс Системс» представляет собой процесс регистрации устройства, которое необходимо взять для работы, сотрудником в веб-приложении.

После добавления администратором сотрудника в систему, он может авторизоваться в ней.

После авторизации сотруднику доступен следующий функционал:

- просмотр списка мобильных устройств;
- изменение статуса мобильных устройств;
- комментирование статуса мобильных устройств;
- выход из системы.

В системе присутствует роль «Администратор», которая включает в себя возможности «Держателя», а также другие возможности такие как:

- добавление, редактирование и удаление информации о сотруднике;
- добавление, редактирование и удаление информации о мобильных устройствах;
- просмотр информации о сотрудниках.

2.3 Разработка информационной модели

Информационный объект – это описание некоторой сущности предметной области реального объекта, процесса, явления или события. Информационный объект образуется совокупностью логически взаимосвязанных реквизитов, представляющих качественные и количественные характеристики сущности.

Проанализировав возможные пользовательские действия была спроектирована информационная модель базы данных, представленная на рисунке 2.2.

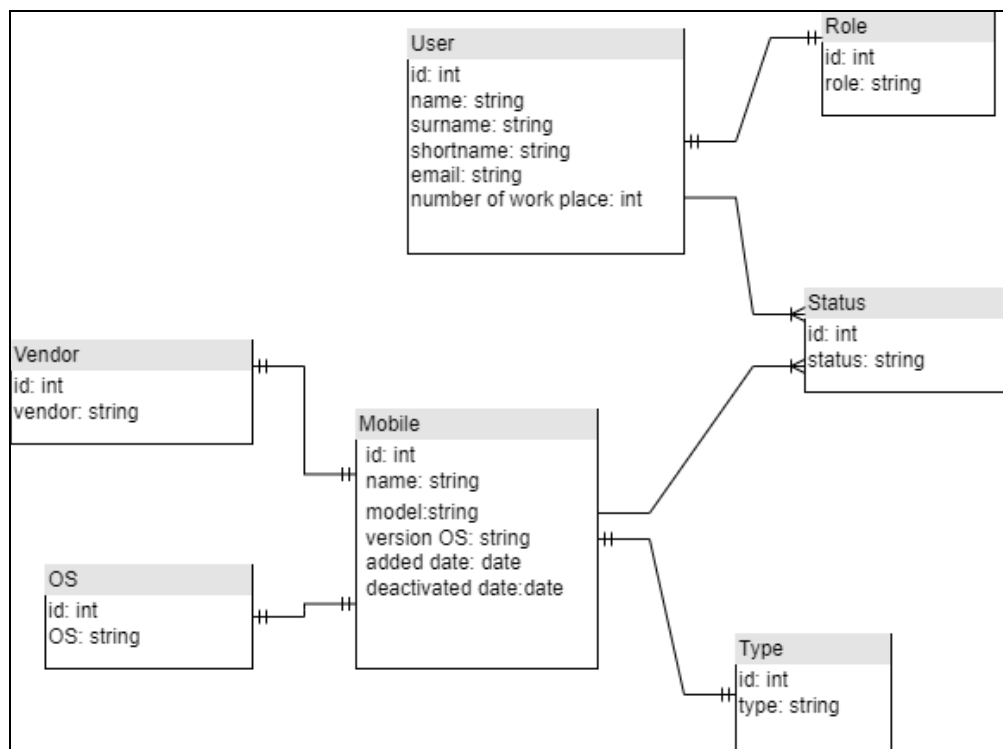


Рисунок 2.2 – Информационная модель БД

В процессе анализа предметной области были выделены следующие типы сущностей:

- User – сущность, хранящая пользовательские учетные записи с требуемой пользовательской информацией. Атрибуты: id – идентификатор сотрудника; name – имя сотрудника; surname – фамилия сотрудника; shortname – сокращенное имя сотрудника; email – почтовый адрес сотрудника; number of work place – номер рабочего места сотрудника;

- Role – сущность для хранения ролей в системе. Атрибуты: id – идентификатор роли; role – наименование роли;

- Mobile – сущность для хранения информации о мобильных устройствах. Атрибуты: id – идентификатор мобильного устройства; name – название мобильного устройства; model – модель мобильного устройства; version OS – версия операционной системы; added date – дата покупки мобильного устройства; deactivated date – дата деактивации мобильного устройства;

- Status – сущность для хранения статуса мобильного устройства. Атрибуты: id – идентификатор статуса мобильного устройства; status – статус мобильного устройства;

- Vendor – сущность для хранения информации о вендоре мобильного устройства. Атрибуты: id – идентификатор вендора мобильного устройства; vendor – наименование вендора мобильного устройства;

- OS – сущность для хранения информации об операционной системе мобильного устройства. Атрибуты: id – идентификатор операционной системы мобильного устройства; OS – наименование операционной системы мобильного устройства;

– Type – сущность для хранения информации о типе устройства.
Атрибуты: id – идентификатор типа устройства; type – тип устройства.

2.4 Разработка модели взаимодействия пользователя с интерфейсом

Для архитектуры был выбран архитектурный шаблон MVC.

Шаблон архитектуры Model-View-Controller (MVC) разделяет приложение на три основных компонента: модель, представление и контроллер. Платформа ASP.NET MVC представляет собой альтернативу схеме веб-форм ASP.NET при создании веб-приложений. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например, с главными страницами и проверкой подлинности на основе членства. Платформа MVC определяется в сборке System.Web.Mvc. Схема шаблона разработки MVC представлен на рисунке 2.3.

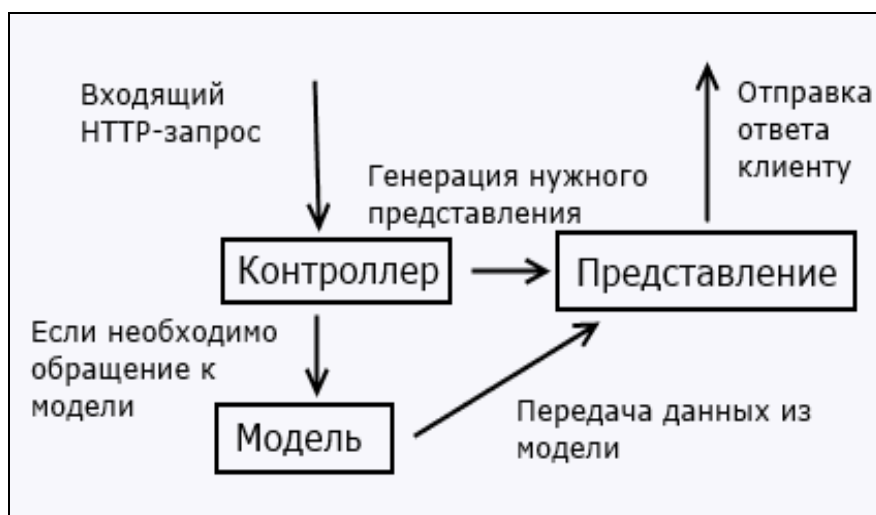


Рисунок 2.3 – Схема шаблона разработки Model View Controller

Шаблон разработки MVC позволяет создавать приложения, различные аспекты которых (логика ввода, бизнес-логика и логика интерфейса) разделены, но достаточно тесно взаимодействуют друг с другом. Эта схема указывает расположение каждого вида логики в приложении. Пользовательский интерфейс располагается в представлении. Логика ввода располагается в контроллере. Бизнес-логика находится в модели. Это разделение позволяет работать со сложными структурами при создании приложения, так как обеспечивает одновременную реализацию только одного аспекта. Например, разработчик может сконцентрироваться на создании представления отдельно от бизнес-логики [11].

Связь между основными компонентами приложения MVC также облегчает параллельную разработку. Например, один разработчик может

создавать представление, другой – логику контроллера, а третий – бизнес-логику модели.

Платформа ASP.NET MVC имеет следующие преимущества:

- облегчает управление сложными структурами путем разделения приложения на модель, представление и контроллер;
- не использует состояние просмотра и серверные формы. Это делает платформу MVC идеальной для разработчиков, которым необходим полный контроль над поведением приложения;
- использует схему основного контроллера, при которой запросы веб-приложения обрабатываются через один контроллер. Это позволяет создавать приложения, поддерживающие расширенную инфраструктуру маршрутизации;
- обеспечивает расширенную поддержку разработки на основе тестирования;
- хорошо подходит для веб-приложений, поддерживаемых крупными коллективами разработчиков, а также веб-разработчикам, которым необходим высокий уровень контроля над поведением приложения [11].

2.5 Разработка технических требований к программному средству

Разрабатываемое программное средство должно обеспечивать корректное функционирование при развертывании программных модулей на сервере со следующими техническими характеристиками:

- Intel 2.2 ГГц или более быстродействующий процессор;
- оперативная память 4 Гбайт или более;
- доступный объем дискового пространства 100 Гбайт.

Для нормального функционирования клиентской части программного средства должны выполняться следующие технические требования:

- Pentium 2 ГГц или более быстродействующий процессор;
 - оперативная память 2 Гбайт или более;
 - 32- или 64-битная версия операционных систем Windows 10, 11;
- браузер Google Chrome версии 95.x.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Обоснование выбора среды разработки

В качестве среды разработки была выбрана Microsoft Visual Studio 2019 Community.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного средства и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного средства (например, клиент Team Explorer для работы с Team Foundation Server) [3].

Для написания данной системы была выбрана технология ASP.NET Core.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

- создавать веб-приложения и службы, приложения IoT и серверные части для мобильных приложений;
- использовать избранные средства разработки в Windows, macOS и Linux;
- выполнять развертывания в облаке или локальной среде;
- работать в .NET Core или .NET Framework [3].

ASP.NET Core предоставляет следующие преимущества:

- единое решение для создания пользовательского веб-интерфейса и веб-API;
- интеграция современных клиентских платформ и рабочих процессов разработки;
- облачная система конфигурации на основе среды;
- встроенное введение зависимостей;
- упрощенный высокопроизводительный модульный конвейер HTTP-запросов;
- возможность размещения в IIS, Nginx, Apache, Docker или в собственном процессе;
- параллельное управление версиями приложения, ориентированное на .NET Core;
- инструментарий, упрощающий процесс современной веб-разработки;
- возможность сборки и запуска в ОС Windows, macOS и Linux;
- открытый исходный код и ориентация на сообщество [3].

ASP.NET Core поставляется полностью в виде пакетов NuGet. Использование пакетов NuGet позволяет оптимизировать приложения для включения только необходимых зависимостей. На самом деле для приложений ASP.NET Core 2.x, ориентированных на .NET Core, требуется только один пакет NuGet. За счет небольшого размера контактной зоны приложения доступны такие преимущества, как более высокий уровень безопасности, минимальное обслуживание и улучшенная производительность [3].

Для написания данной системы был выбран язык программирования C#.

C# – это изящный объектно-ориентированный язык со строгой типизацией, позволяющий разработчикам создавать различные безопасные и надежные приложения, работающие на платформе .NET Framework. C# можно использовать для создания клиентских приложений Windows, XML-веб-служб, распределенных компонентов, приложений клиент-сервер, приложений баз данных и т. д. Visual C# предоставляет развитый редактор кода, удобные конструкторы пользовательского интерфейса, интегрированный отладчик и многие другие средства, которые упрощают разработку приложений на языке C# для платформы .NET Framework.

Помимо этого, C# предлагает ряд инновационных языковых конструкций, упрощающих разработку программных компонентов. Инкапсулированные сигнатуры методов, именуемые делегатами, которые позволяют реализовать типобезопасные уведомления о событиях. Свойства, выполняющие функцию аксессоров для закрытых переменных-членов. Атрибуты, предоставляющие декларативные метаданные о типах во время выполнения. Внутри строчные комментарии для XML-документации. LINQ для создания запросов к различным источникам данных [12].

Так же при разработке данного программного средства были задействованы: HTML, CSS, Bootstrap.

HTML – язык разметки (маркировки) гипертекста. HTML дает возможность производить переход от одной части текста к другой, и, что замечательно, эти части могут храниться на совершенно разных компьютерах. Он создан специально для разметки веб-страниц. Именно язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе.

CSS – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Bootstrap – это инструментарий с открытым исходным кодом для разработки с помощью HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JS (JavaScript). Он включает в себя множество разных компонентов для веб-сайта: типографику, веб-формы, кнопки, блоки навигации и другие. Основные преимущества Bootstrap:

- адаптивность – дизайн сайта будет корректно отображаться на экранах устройств разных размеров вне зависимости от диагонали;
- кроссбраузерность – одинаковое отображение во всех браузерах;
- легкость в использовании;
- понятный код – позволяет писать качественный и понятный код, что облегчает чтение кода для других разработчиков [12].

Для работы с БД использовался MS SQL Server.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основным используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

MS SQL Server характеризуется такими особенностями как:

- производительность. MS SQL Server работает очень быстро;
- надежность и безопасность. MS SQL Server предоставляет шифрование данных;
- простота. С данной СУБД относительно легко работать и вести администрирование [12].

При создании ПС использовалась технология Razor Pages.

Razor Pages предоставляет технологию, альтернативную системе Model-View-Controller. Razor Pages позволяют создавать страницы с кодом Razor, которые могут обрабатывать запросы. В некоторой степени эта

функциональности напоминает работу веб-форм, которые представляли страницу с расширением aspx и имели файл логики на C#, связанный с данной страницей. В этом плане Razor Pages представляют альтернативу стандартной модели MVC для построения приложения [12].

3.2 Разработка структурной схемы программного средства

В ходе моделирования было определено, что для разработки структурной схемы программного средства будет использоваться архитектура MVC.

MVC представляет собой стандартный шаблон разработки, знакомый многим специалистам. Некоторые типы веб-приложений имеют преимущества при создании на платформе MVC. Для других может быть целесообразно использование традиционной схемы приложения ASP.NET, основанной на веб-формах и обратной передаче. В некоторых случаях возможно сочетание двух подходов: применение одной схемы не исключает использования другой [8].

В состав платформы MVC входят следующие компоненты.

Модели являются частями приложения, реализующими логику для домена данных приложения. Объекты моделей часто получают и сохраняют состояние модели в базе данных. Например, объект Product может получать информацию из базы данных, работать с ней, а затем записывать обновленные данные в таблицу Products базы данных SQL Server [8].

Представления служат для отображения пользовательского интерфейса приложения. Пользовательский интерфейс обычно создается на основе данных модели. Примером может служить представление для редактирования таблицы Products, которое содержит текстовые поля, раскрывающиеся списки и флажки, значения которых основаны на текущем состоянии объекта Product [8].

Контроллеры осуществляют взаимодействие с пользователем, работу с моделью, а также выбор представления, отображающего пользовательский интерфейс. В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных [8].

На рисунке 3.1 показана архитектура MVC приложения.

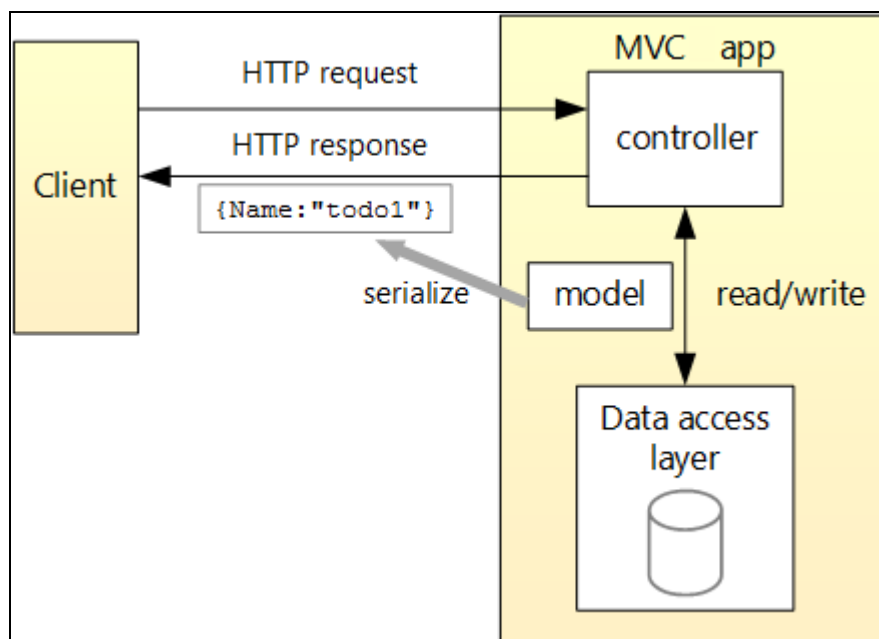


Рисунок 3.1 – архитектура MVC приложения

3.3 Разработка структуры классов

Диаграмма классов (class diagram) – основной способ описания структуры системы. На диаграмме классов применяется один основной тип сущностей: классы (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений:

- ассоциация между классами;
- обобщение между классами;
- зависимости (различных типов) между классами и между классами и интерфейсами.

На рисунке 3.2 представлена диаграмма классов уровня контроллера.

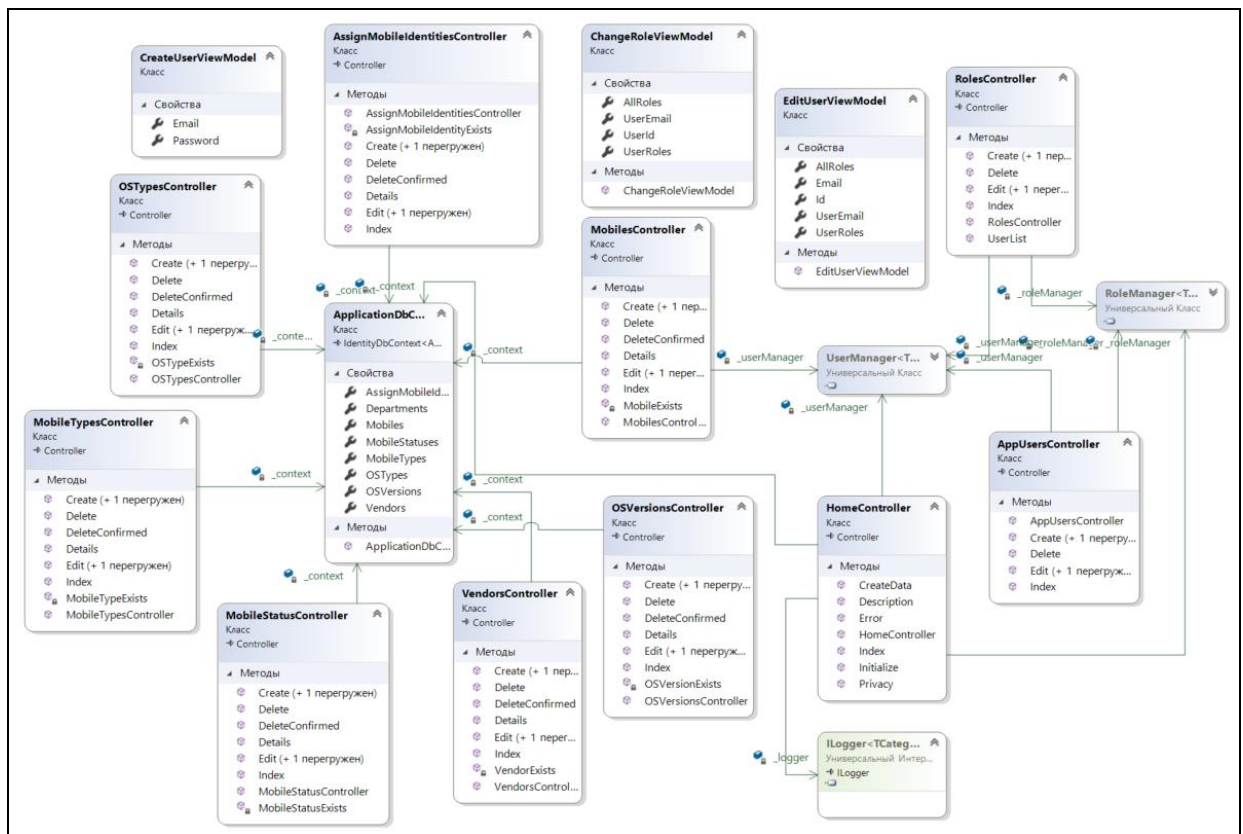


Рисунок 3.2 – Диаграмма классов уровня контроллера

Диаграмма классов уровня модели данных представлена на рисунке 3.3.

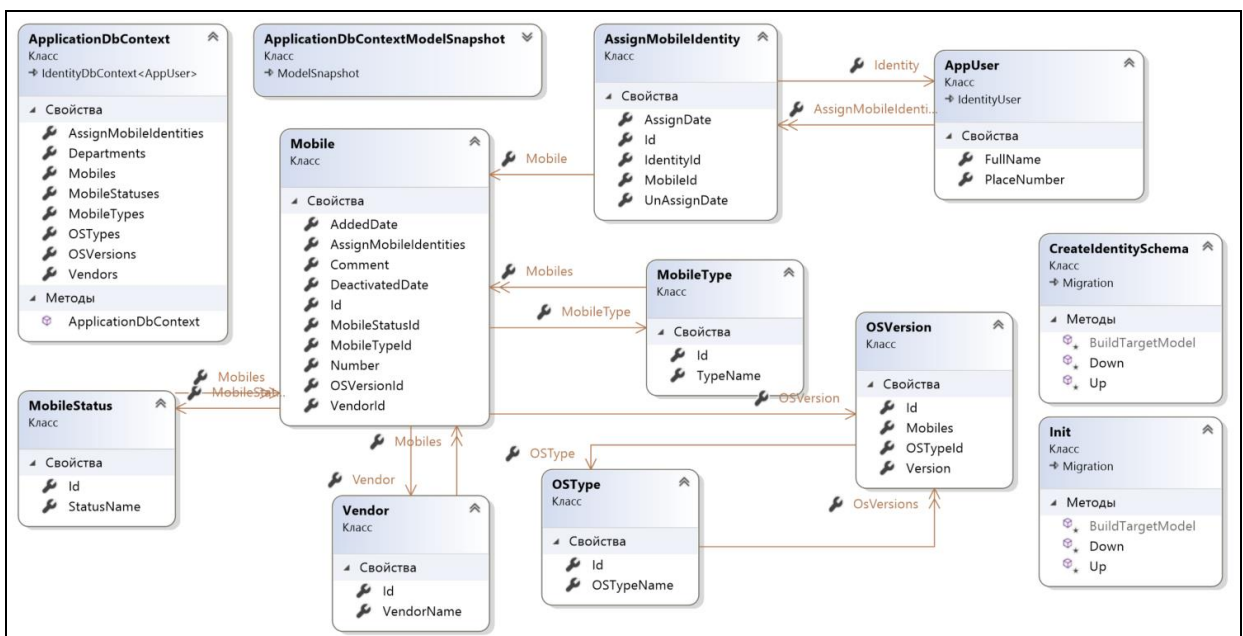


Рисунок 3.3 – Диаграмма классов уровня модели данных

3.4 Разработка физической модели данных

На рисунке 3.4 представлена физическая модель базы данных, построенная на основе информационной модели базы данных. Данная модель жестко связана с конкретной системой управления базами данных. В данной модели представлены реальные типы данных, ограничения, реальные названия таблиц базы данных.

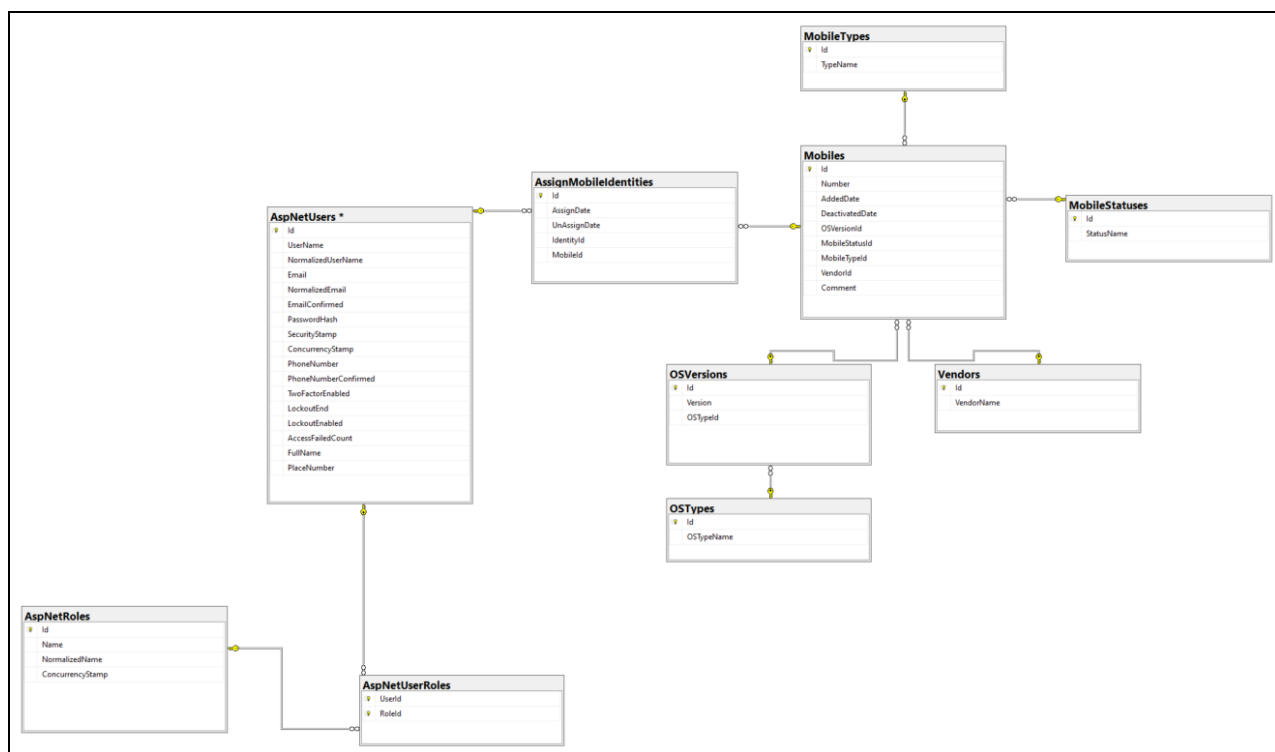


Рисунок 3.4 – Физическая модель БД

Формализованное описание объектов представлено в таблицах 3.1–3.10. Описание таблицы «AspNetUsers» представлено в таблице 3.1.

Таблица 3.1 – Описание таблицы «AspNetUsers»

Поле	Тип данных	Описание
1	2	3
Id	nvarchar(450)	Идентификатор сотрудника
UserName	nvarchar(256)	Имя сотрудника
NormalizedUserName	nvarchar(256)	Нормализованное имя сотрудника
Email	nvarchar(256)	Почтовый адрес сотрудника
NormalizedEmail	nvarchar(256)	Нормализованный почтовый адрес сотрудника
EmailConfirmed	bit	Флаг подтверждения почтового адреса сотрудника
PasswordHash	nvarchar(MAX)	Хеш пароля сотрудника

Продолжение таблицы 3.1

1	2	3
SecurityStamp	nvarchar(MAX)	Метка безопасности
ConcurrencyStamp	nvarchar(MAX)	Метка параллелизма
PhoneNumber	nvarchar(MAX)	Мобильный номер сотрудника
PhoneNumberConfirmed	bit	Флаг подтверждения мобильного номера сотрудника
TwoFactorEnabled	bit	Флаг двухфакторной аутентификации
LockoutEnd	datetimeoffset(7)	Дата окончания запрета авторизации сотрудника
LockoutEnabled	bit	Флаг запрета авторизации сотрудника
AccessFailedCount	int	Число попыток неудачного входа в систему
FullName	nvarchar(MAX)	Полное имя сотрудника
PlaceNumber	nvarchar(MAX)	Рабочее место сотрудника

Описание таблицы «Mobiles» представлено в таблице 3.2.

Таблица 3.2 – Описание таблицы «Mobiles»

Поле	Тип данных	Описание
Id	int	Идентификатор мобильного устройства
Number	nvarchar(MAX)	Краткий идентификатор мобильного устройства
AddedDate	datetime2(7)	Дата покупки мобильного устройства
DeactivatedDate	datetime2(7)	Дата деактивации мобильного устройства
OSVersionId	int	Внешний ключ идентификатора версии операционной системы
MobileStatusId	int	Внешний ключ идентификатора статуса мобильного устройства
MobileTypeId	int	Внешний ключ типа мобильного устройства
VendorId	int	Внешний ключ вендора мобильного устройства
Comment	nvarchar(MAX)	Комментарий

Описание таблицы «AspNetRoles» представлено в таблице 3.3.

Таблица 3.3 – Описание таблицы «AspNetRoles»

Поле	Тип данных	Описание
Id	nvarchar(450)	Идентификатор роли сотрудника
Name	nvarchar(256)	Наименование роли сотрудника
NormalizedName	nvarchar(256)	Нормализованное наименование роли сотрудника
ConcurrencyStamp	nvarchar(MAX)	Метка параллелизма

Описание таблицы «AspNetUserRoles» представлено в таблице 3.4.

Таблица 3.4 – Описание таблицы «AspNetUserRoles»

Поле	Тип данных	Описание
UserId	nvarchar(450)	Идентификатор сотрудника
RoleId	nvarchar(450)	Идентификатор роли сотрудника

Описание таблицы «Vendors» представлено в таблице 3.5.

Таблица 3.5 – Описание таблицы «Vendors»

Поле	Тип данных	Описание
Id	int	Идентификатор вендора
VendorName	nvarchar(MAX)	Наименование вендора

Описание таблицы «MobileStatuses» представлено в таблице 3.6.

Таблица 3.6 – Описание таблицы «MobileStatuses»

Поле	Тип данных	Описание
Id	int	Идентификатор статуса мобильного устройства
StatusName	nvarchar(MAX)	Наименование статуса мобильного устройства

Описание таблицы «MobileTypes» представлено в таблице 3.7.

Таблица 3.7 – Описание таблицы «MobileTypes»

Поле	Тип данных	Описание
Id	int	Идентификатор типа мобильного устройства
TypeName	nvarchar(MAX)	Наименование типа мобильного устройства

Описание таблицы «OSTypes» представлено в таблице 3.8.

Таблица 3.8 – Описание таблицы «OSTypes»

Поле	Тип данных	Описание
Id	int	Идентификатор типа операционной системы
OSTypeName	nvarchar(MAX)	Наименование типа операционной системы

Описание таблицы «OSVersions» представлено в таблице 3.9.

Таблица 3.9 – Описание таблицы «OSVersions»

Поле	Тип данных	Описание
Id	int	Идентификатор версии операционной системы
Version	nvarchar(MAX)	Версия операционной системы
OSTypeId	int	Внешний ключ идентификатора типа операционной системы

Описание таблицы «AssignMobileIdentities» представлено в таблице 3.10.

Таблица 3.10 – Описание таблицы «AssignMobileIdentities»

Поле	Тип данных	Описание
Id	int	Идентификатор
AssignDate	datetime2(7)	Дата назначения мобильного устройства на сотрудника
UnAssignDate	datetime2(7)	Дата снятия мобильного устройства с сотрудника
IdentityId	int	Внешний ключ идентификатора сотрудника

Продолжение таблицы 3.10

MobileId	int	Внешний идентификатор мобильного устройства	ключ
----------	-----	---	------

3.5 Проектирование алгоритмов работы программного средства

Для определения логики написания программы были спроектированы следующие алгоритмы работы программного средства.

3.5.1 Алгоритм проверки уровня доступа на стороне сервера

Алгоритм проверки уровня доступа на стороне сервера представлен на рисунке 3.5. Данный алгоритм демонстрирует, как происходит проверка уровня доступа пользователя при отправке запроса на сторону сервера.

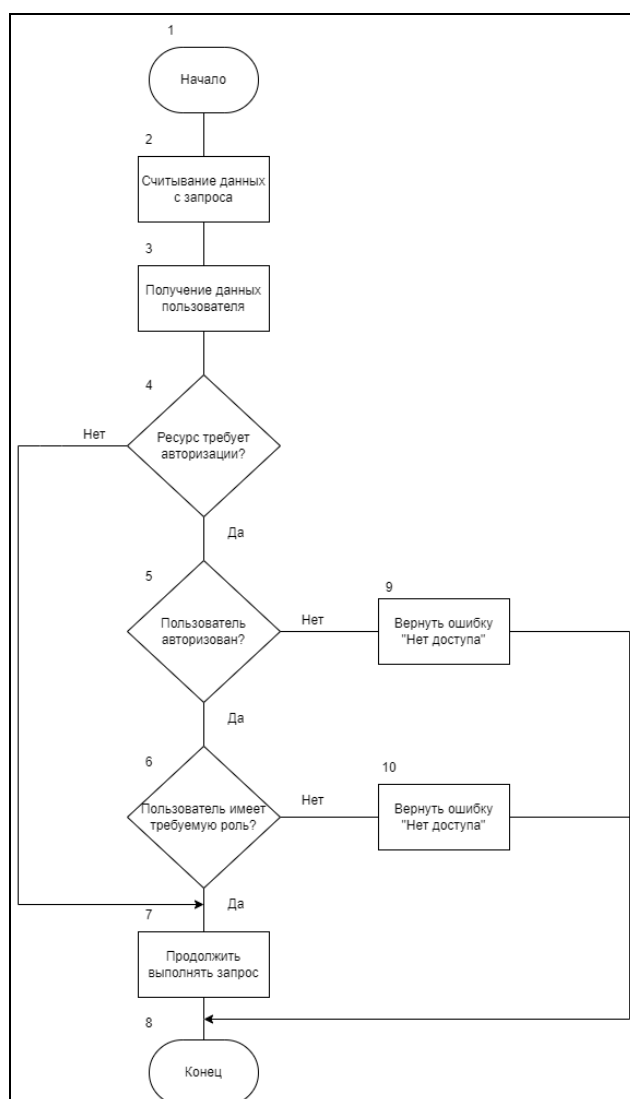


Рисунок 3.5 – Блок-схема алгоритма проверки уровня доступа на стороне сервера

3.5.2 Алгоритм добавления сотрудника

Алгоритм добавления сотрудника в список представлен на рисунке 3.6. Данный алгоритм демонстрирует, как происходит добавление сотрудника.

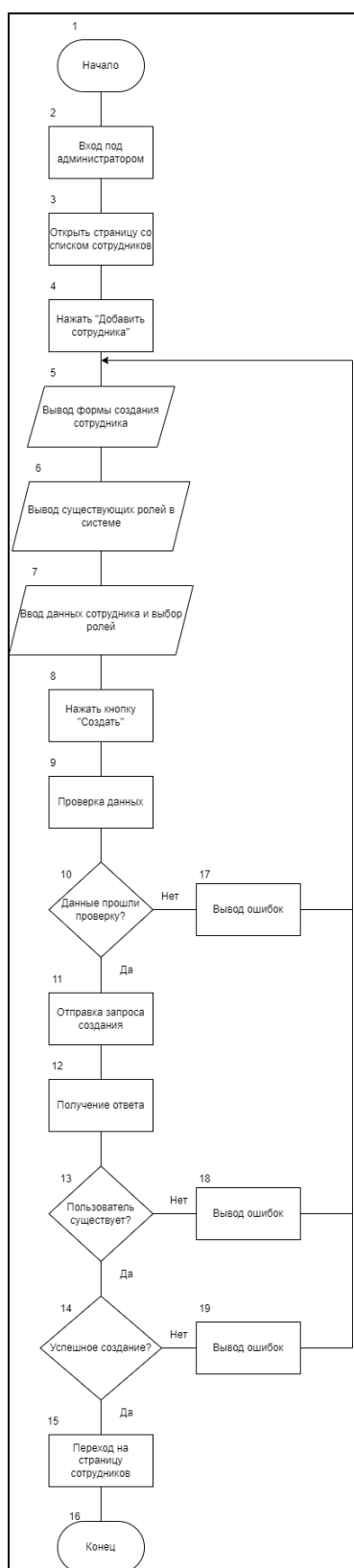


Рисунок 3.6 – Блок-схема алгоритма добавления сотрудника в список

3.5.3 Алгоритм добавления устройства

Алгоритм добавления устройства в список представлен на рисунке 3.7. Данный алгоритм демонстрирует, как происходит добавление устройства.

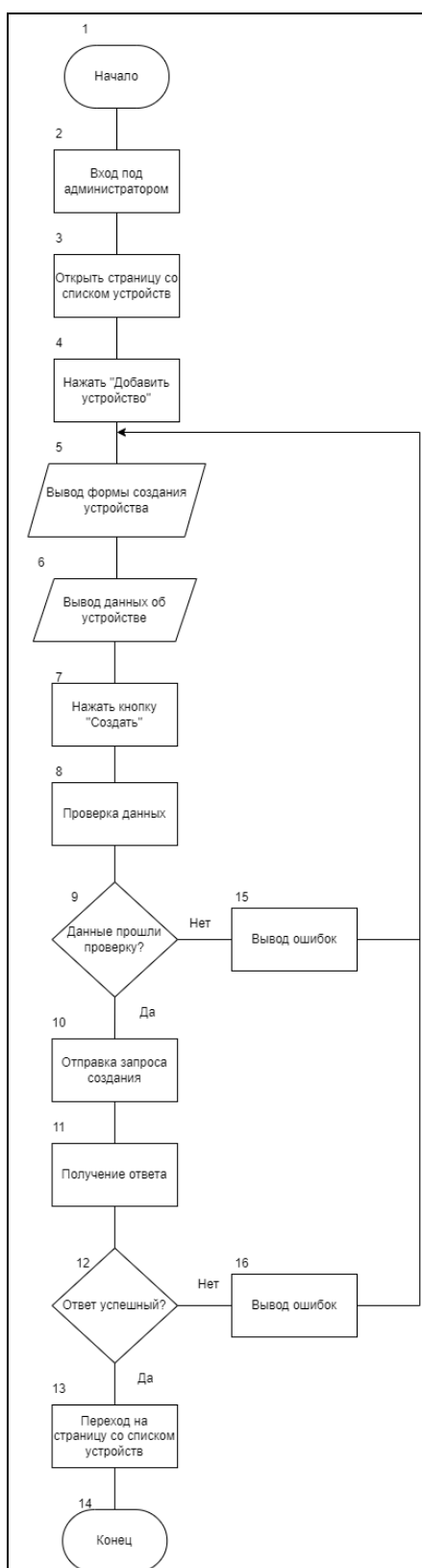


Рисунок 3.7 – Блок-схема алгоритма добавления устройства в список

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства – это процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

Существующие на сегодня методы тестирования ПС не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПС.

Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода. То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла ПС.

Существуют следующие виды тестирования по целям и задачам:

- функциональное;
- нефункциональное.

Функциональное тестирование – это тестирование программного средства, основанное на анализе спецификации функциональности приложения, проводимое с целью проверки на соответствие требованиям, то есть проверка того, «что» система делает.

Нефункциональное тестирование – это тестирование свойств программного средства, которые не относятся к функциональности системы, то есть проверка того, «как» система работает (надежность, эффективность, практичность, сопровождаемость).

В рамках тестирования ПС будет проведено функциональное тестирование с использованием тест-кейсов.

Тест-кейс – это набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Тест-кейс №1 для проверки работы веб-приложения в различных браузерах представлен в таблице 4.1.

Таблица 4.1 – Тест-кейс №1

Тест	Ожидаемый результат	Результат
Запуск веб-приложения в различных браузерах 1. Запустить веб-приложение в браузере Google Chrome 2. Запустить веб-приложение в браузере Mozilla Firefox	1. Веб-приложение успешно запущено в браузере Google Chrome 2. Веб-приложение успешно запущено в браузере Mozilla Firefox	Успех

Тест-кейс №2 для проверки наличия всех элементов главной страницы представлен в таблице 4.2.

Таблица 4.2 – Тест-кейс №2

Тест	Ожидаемый результат	Результат
Наличие элементов главной страницы 1. Запустить веб-приложение	1. Отображаются элементы главной страницы: - название веб-приложения «Учет мобильных устройств» - вкладка «О программе» - вкладка «Вход» (открыта по умолчанию) - нижняя панель	Успех

Тест-кейс №3 для проверки авторизации пользователя представлен в таблице 4.3.

Таблица 4.3 – Тест-кейс №3

Тест	Ожидаемый результат	Результат
Авторизация пользователя 1. Запустить веб-приложение 2. Ввести валидные логин и пароль 3. Нажать кнопку «Войти»	1. Веб-приложение запущено 2. Поля «Логин» и «Пароль» заполнены, данные в поле «Пароль» замаскированы 3. Отображается главная страница авторизованной части приложения	Успех

Тест-кейс №4 для проверки отображения сообщения об ошибке при авторизации пользователя, когда введены некорректные данные для входа в систему, представлен в таблице 4.4.

Таблица 4.4 – Тест-кейс №4

Тест	Ожидаемый результат	Результат
Отображение сообщения об ошибке при авторизации пользователя, когда введены некорректные данные для входа в систему 1. Запустить веб-приложение 2. Ввести невалидные логин и пароль 3. Нажать кнопку «Войти»	1. Веб-приложение запущено 2. Поля «Логин» и «Пароль» заполнены, данные в поле «Пароль» замаскированы 3. Отображается сообщение об ошибке «Введены некорректные данные для входа в систему»	Успех

Тест-кейс №5 для проверки изменения статуса мобильного устройства пользователем представлен в таблице 4.5.

Таблица 4.5 – Тест-кейс №5

Тест	Ожидаемый результат	Результат
Изменение статуса мобильного устройства пользователем 1. Авторизоваться 2. Нажать кнопку «Изменить статус» напротив выбранного устройства 3. Выбрать в поле «Статус» значение «Взят» 4. Нажать кнопку «Сохранить»	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Изменить статус устройства» 3. В поле «Статус» отображается значение «Взят» 4. В информации об устройстве отображается информация о сотруднике, который его взял (Email, ФИО, Номер рабочего места). Меняется статус устройства на «Взят»	Успех

Тест-кейс №6 для проверки просмотра детальной информации об устройстве представлен в таблице 4.6.

Таблица 4.6 – Тест-кейс №6

Тест	Ожидаемый результат	Результат
Просмотра детальной информации об устройстве 1. Авторизоваться 2. Нажать кнопку «Детальная форма» напротив выбранного устройства	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Детальная информация об устройстве»	Успех

Тест-кейс №7 для проверки добавления устройства в список представлен в таблице 4.7.

Таблица 4.7 – Тест-кейс №7

Тест	Ожидаемый результат	Результат
Добавление устройства 1. Авторизоваться под администратором 2. Нажать кнопку «Добавить новое устройство» 3. Заполнить все поля валидными данными 4. Нажать кнопку «Создать»	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Добавление устройства» 3. Все поля заполнены 4. Устройство добавлено в список	Успех

Тест-кейс №8 для проверки редактирования информации об устройстве представлен в таблице 4.8.

Таблица 4.8 – Тест-кейс №8

Тест	Ожидаемый результат	Результат
1	2	3
Редактирование информации об устройстве 1. Авторизоваться под администратором 2. Нажать кнопку «Редактировать» напротив выбранного устройства	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Редактирование устройства»	Успех

Продолжение таблицы 4.8

1	2	3
3. Изменить информацию об устройстве. Например, изменить статус на «Ремонтируется» 4. Нажать кнопку «Сохранить»	3. В поле «Статус» отображается значение «Ремонтируется» 4. В списке устройств обновлена информация об устройстве	

Тест-кейс №9 для проверки добавления сотрудника в список представлен в таблице 4.9.

Таблица 4.9 – Тест-кейс №9

Тест	Ожидаемый результат	Результат
Добавление сотрудника 1. Авторизоваться под администратором 2. Перейти на вкладку «Сотрудники» 3. Нажать кнопку «Добавить сотрудника» 4. Заполнить все поля валидными данными 5. Нажать кнопку «Создать»	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Сотрудники» 3. Отображается страница «Добавление сотрудника» 4. Все поля заполнены Сотрудник добавлен в список	Успех

Тест-кейс №10 для проверки редактирования информации о сотруднике представлен в таблице 4.10.

Таблица 4.10 – Тест-кейс №10

Тест	Ожидаемый результат	Результат
1	2	3
Редактирование информации о сотруднике 1. Авторизоваться под администратором 2. Перейти на вкладку «Сотрудники» 3. Нажать кнопку «Изменить» напротив выбранного сотрудника	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Сотрудники» 3. Отображается страница «Редактирование	Успех

Продолжение таблицы 4.10

1	2	3
4. Изменить информацию о сотруднике. Например, номер рабочего места изменить на 18 5. Нажать кнопку «Сохранить»	информации о сотруднике» 4. В поле «Номер места» отображается значение «18» 5. В списке сотрудников обновлена информация о сотруднике	

Тест-кейс №11 для проверки удаления мобильного устройства из списка представлен в таблице 4.11.

Таблица 4.11 – Тест-кейс №11

Тест	Ожидаемый результат	Результат
Удаление мобильного устройства из списка 1. Авторизоваться под администратором 2. Нажать кнопку «Удалить» напротив выбранного мобильного устройства	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Информация о мобильном устройстве удалена из списка	Успех

Тест-кейс №12 для проверки удаления сотрудника из списка представлен в таблице 4.12.

Таблица 4.12 – Тест-кейс №12

Тест	Ожидаемый результат	Результат
Удаление сотрудника из списка 1. Авторизоваться под администратором 2. Перейти на вкладку «Сотрудники» 3. Нажать кнопку «Удалить» напротив выбранного сотрудника	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается страница «Сотрудники» 3. Информация о сотруднике удалена из списка	Успех

Тест-кейс №13 для проверки выхода пользователя из системы представлен в таблице 4.13.

Таблица 4.13 – Тест-кейс №13

Тест	Ожидаемый результат	Результат
Выход из системы 1. Авторизоваться 2. Нажать кнопку «Выйти»	1. Отображается главная страница авторизованной части приложения, вкладка «Устройства» открыта по умолчанию 2. Отображается главная страница неавторизованной части приложения	Успех

Тест-кейс №14 для проверки перехода к странице «О программе» представлен в таблице 4.14.

Таблица 4.14 – Тест-кейс №14

Тест	Ожидаемый результат	Результат
Выход из системы 1. Запустить веб-приложение 2. Выбрать вкладку «О программе»	1. Веб-приложение запущено 2. Отображается страница «О программе»	Успех

Как видно из результатов функционального тестирования, веб-приложение работает исправно.

5 МЕТОДИКА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

После запуска программного средства открывается главная страница неавторизованной части приложения, которая содержит форму логина и форму с информацией о программе. Данная страница доступна для всех типов пользователей. Главная страница веб-приложения представлена на рисунке 5.1.

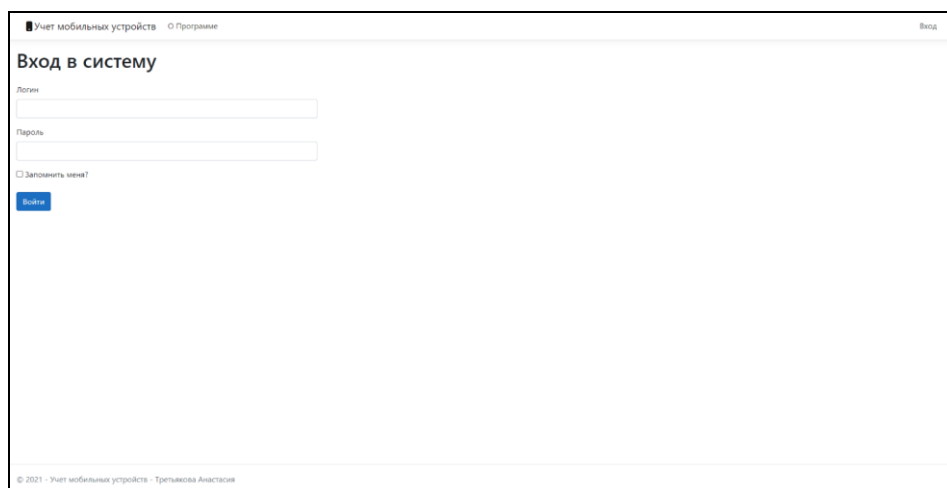


Рисунок 5.1 – Главная страница веб-приложения

5.1 Авторизация пользователя

Для того чтобы пользователь авторизовался, требуется наличие учетной записи в системе. Если пользователь имеет учетную запись требуется в панели навигации нажать на элемент управления «Вход», данная форма открыта по умолчанию. Форма входа представлена на рисунке 5.1.

На форме присутствует проверка ввода данных. Если пользователь не введет данные в одно из полей, появится уведомление, что требуется ввести данные. Пример проверки ввода данных представлен на рисунке 5.2.

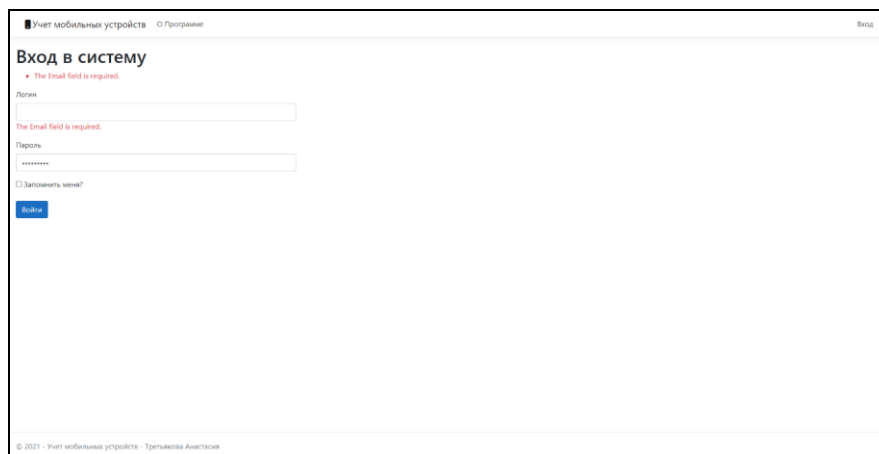


Рисунок 5.2 – Форма входа веб-приложения с указанием ошибки

На форму входа в приложении можно попасть с формы «О программе», выбрав пункт меню «Вход». Форма «О программе» представлена на рисунке 5.3.

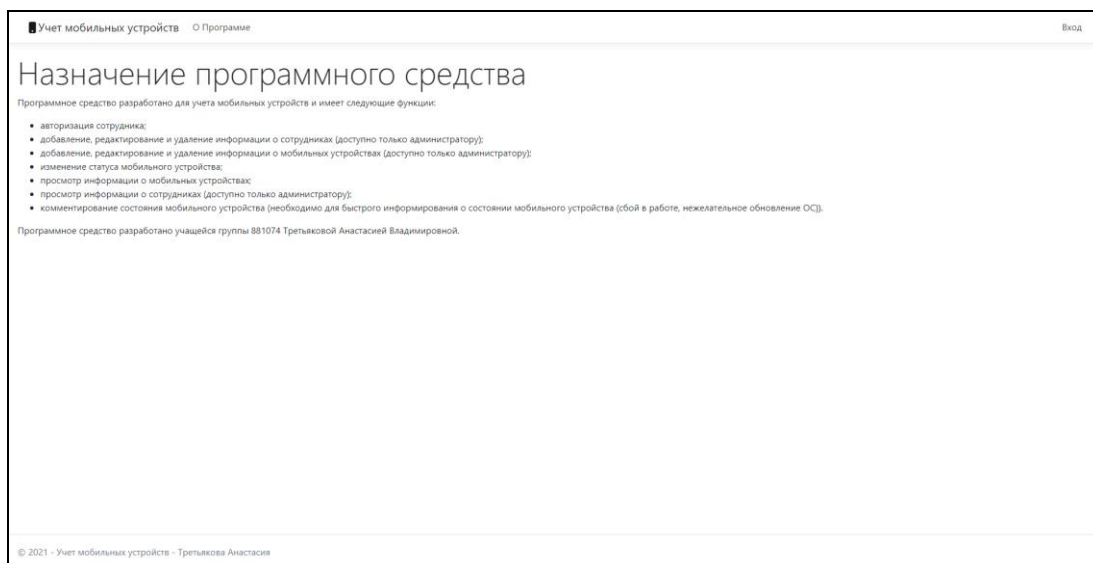


Рисунок 5.3 – Форма «О программе»

После успешной авторизации в веб-приложении, пользователя перенаправляет на страницу со списком устройств. После входа навигационная панель меняет свои элементы. Главная страница с измененной навигационной панелью для авторизованного пользователя представлена на рисунке 5.4.

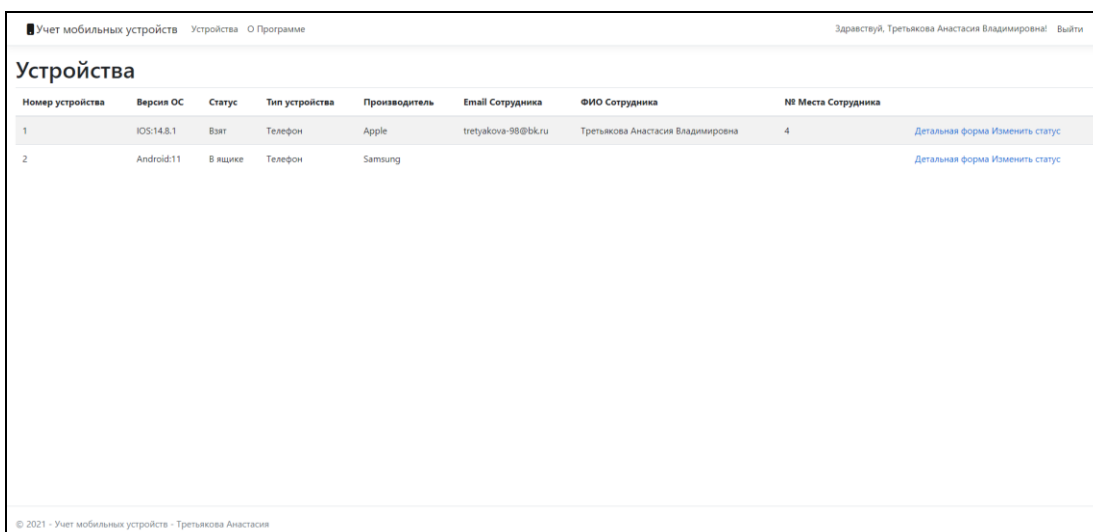


Рисунок 5.3 – Форма «Устройства»

Для выхода из учетной записи требуется нажать на элемент управления «Выйти». После нажатия на данный элемент управления отображается главная страница неавторизованной части приложения.

5.2 Детальная информация об устройстве и его статус

После авторизации в системе пользователю будут доступны возможности для управления мобильными устройствами. Пользователь имеет возможность просмотреть детальную информацию о мобильном устройстве, а также изменить его статус.

Для просмотра детальной информации о мобильном устройстве необходимо нажать на кнопку «Детальная форма». После нажатия на кнопку «Детальная форма» отобразится данная форма со следующей информацией о мобильном устройстве:

- Номер устройства;
- Дата добавления;
- Дата деактивации;
- Версия ОС;
- Статус;
- Email Сотрудника;
- ФИО Сотрудника;
- № Места Сотрудника;
- Тип устройства;
- Производитель;
- Комментарий (смотрите рисунок 5.4).

Учет мобильных устройств		Устройства	О Программе	Здравствуй, Третьякова Анастасия Владимировна! Выйти	
Устройство					
Номер устройства	1				
Дата добавления	01.01.0001 00:00:00				
Дата деактивации					
Версия ОС	IOS:14.8.1				
Статус	Взят				
Email Сотрудника	treyakova-98@bk.ru				
ФИО Сотрудника	Третьякова Анастасия Владимировна				
№ Места Сотрудника	4				
Тип устройства	Телефон				
Производитель	Apple				
Комментарий					
Редактировать Вернуться к списку					
© 2021 - Учет мобильных устройств - Третьякова Анастасия					

Рисунок 5.4 – Форма «Детальная форма»

Также на данной форме есть кнопки «Редактировать» и «Вернуться к списку». По нажатию на кнопку «Редактировать» пользователь переходит к форме «Изменить статус». По нажатию на кнопку «Вернуться к списку» пользователь переходит к форме «Устройства».

Также для перехода к форме «Изменить статус» можно нажать на кнопку «Изменить статус», рисунок данной формы смотрите на рисунке 5.5.

Учет мобильных устройств Устройства О Программе Здравствуй, Третьякова Анастасия Владимировна! Выйти

Устройство

Статус
Взят

Комментарий

[Сохранить](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.5 – Форма «Изменить статус»

На данной форме пользователь может изменить статус устройства и оставить комментарий к выбранному мобильному устройству при необходимости. По нажатию на кнопку «Сохранить» пользователь будет попадать на обновленную страницу «Устройства». По нажатию на кнопку «Вернуться к списку» пользователь также будет перемещен на форму «Устройства», введенная им информация не будет сохранена.

5.3 Администратор

В веб-приложении существует пользователь с ролью администратор, ему доступны все вышеперечисленные возможности системы, а также возможность редактировать и удалять данные о мобильных устройствах, возможность добавлять, изменять и удалять данные о сотрудниках. После авторизации в системе администратор попадает на форму «Устройства», смотрите рисунок 5.6.

Учет мобильных устройств Устройства Данные Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Устройства

[Добавить новое устройство](#)

Номер устройства	Версия ОС	Статус	Тип устройства	Производитель	Email Сотрудника	ФИО Сотрудника	№ Места Сотрудника	
1	IOS:14.8.1	Взят	Телефон	Apple	tretyakova-98@bk.ru	Третьякова Анастасия Владимировна	4	Редактировать Детальная форма Удалить
2	Android:11	В ящике	Телефон	Samsung				Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.6 – Форма «Устройства»

Для выхода из учетной записи требуется нажать на элемент управления «Выйти». После нажатия на данный элемент управления отображается главная страница неавторизованной части приложения.

По нажатию на кнопку «Добавить новое устройство» администратор попадает на соответствующую страницу, смотрите рисунок 5.7.

Учет мобильных устройств | Устройства | Данные ▾ | Сотрудники | Роли | О Программе | Здравствуй, Администратор! | Выйти

Устройство

Номер устройства

Дата добавления

Дата деактивации

Версия ОС

Статус

Тип устройства

Производитель

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.7 – Форма «Добавление устройства»

Для заполнения данных на этой форме необходимо, чтобы были внесены данные по типу ОС, версии ОС, статусу, типу устройства, производителям. После внесения данных на форме для их сохранения необходимо нажать кнопку «Сохранить», если администратору необходимо вернуться на форму «Устройства» без сохранения, ему необходимо нажать на кнопку «Вернуться к списку».

Для внесения информации о типах ОС администратору необходимо выбрать в панели навигации «Данные», а затем выбрать в пункт меню «Тип ОС», смотрите рисунок 5.8.

Учет мобильных устройств | Устройства | Данные ▾ | Сотрудники | Роли | О Программе | Здравствуй, Администратор! | Выйти

Типы ОС

[Добавить тип ОС](#)

Тип ОС	
Android	Редактировать Детальная форма Удалить
IOS	Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.8 – Форма «Типы ОС»

Для добавления типа ОС необходимо нажать на кнопку «Добавить тип ОС», после чего отобразится соответствующая форма, смотрите рисунок 5.9.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Тип ОС

Тип ОС

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.9 – Форма «Добавление типа ОС»

После сохранения данные отобразятся на странице «Типы ОС», также данную информацию можно редактировать, удалять, можно просмотреть детальную информацию по типу ОС.

Для внесения информации о версиях ОС администратору необходимо выбрать в панели навигации «Данные», а затем выбрать в пункт меню «Версии устройства», смотрите рисунок 5.10.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Версии ОС

[Добавить новую версию](#)

Версия	Тип ОС	
11	Android	Редактировать Детальная форма Удалить
14.8.1	IOS	Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.10 – Форма «Версии ОС»

Для добавления версии ОС необходимо нажать на кнопку «Добавить новую версию», после чего отобразится соответствующая форма, смотрите рисунок 5.11.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Версия ОС

Версия

Тип ОС

Android ▾

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.11 – Форма «Добавление версии ОС»

После сохранения данные отобразятся на странице «Версии ОС», также данную информацию можно редактировать, удалять, можно просмотреть детальную информацию по версии ОС.

Для внесения информации о типах устройств администратору необходимо выбрать в панели навигации «Данные», а затем выбрать в пункт меню «Тип устройства», смотрите рисунок 5.12.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Типы устройств

[Добавить новый тип устройства](#)

Тип устройства	
Телефон	Редактировать Детальная форма Удалить
Планшет	Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.12 – Форма «Типы устройств»

Для добавления типа устройства необходимо нажать на кнопку «Добавить новый тип устройства», после чего отобразится соответствующая форма, смотрите рисунок 5.13.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Тип устройства

Тип устройства

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.13 – Форма «Добавление типа устройства»

После сохранения данные отобразятся на странице «Типы устройств», также данную информацию можно редактировать, удалять, можно просмотреть детальную информацию по типу устройства.

Для внесения информации о статусах устройств администратору необходимо выбрать в панели навигации «Данные», а затем выбрать в пункт меню «Статусы устройства», смотрите рисунок 5.14.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Статус устройств

[Добавить статус устройства](#)

Статус	
Взят	Редактировать Детальная форма Удалить
В ящике	Редактировать Детальная форма Удалить
Ремонтируется	Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.14 – Форма «Статус устройств»

Для добавления статуса устройства необходимо нажать на кнопку «Добавить статус устройства», после чего отобразится соответствующая форма, смотрите рисунок 5.15.

Учет мобильных устройств Устройства Данные Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Статус устройства

Статус

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.15 – Форма «Добавление статуса устройства»

После сохранения данные отобразятся на странице «Статус устройств», также данную информацию можно редактировать, удалять, можно просмотреть детальную информацию по статусу устройства.

Для внесения информации о мобильных поставщиках администратору необходимо выбрать в панели навигации «Данные», а затем выбрать в пункт меню «Компании», смотрите рисунок 5.16.

Учет мобильных устройств Устройства Данные Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Мобильный поставщик

[Добавить поставщика](#)

Поставщи	
Apple	Редактировать Детальная форма Удалить
Samsung	Редактировать Детальная форма Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.16 – Форма «Мобильный поставщик»

Для добавления информации о поставщике необходимо нажать на кнопку «Добавить поставщика», после чего отобразится соответствующая форма, смотрите рисунок 5.17.

Учет мобильных устройств | Устройства | Данные ▾ | Сотрудники | Роли | О Программе | Здравствуй, Администратор! | Выйти

Мобильный поставщик

Поставщик

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.17 – Форма «Добавление мобильного поставщика»

После сохранения данные отобразятся на странице «Мобильный поставщик», также данную информацию можно редактировать, удалять, можно просмотреть детальную информацию по поставщику мобильного устройства.

Администратору доступно создание и удаление ролей в системе. Форма «Список ролей» представлена на рисунке 5.18.

Учет мобильных устройств | Устройства | Данные ▾ | Сотрудники | Роли | О Программе | Здравствуй, Администратор! | Выйти

Список ролей

[Добавить роль](#)

Наименование	
Администратор	Удалить
Пользователь	Удалить

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.18 – Форма «Список ролей»

Администратору доступно добавление, редактирование и удаление сотрудников, страница «Сотрудники» представлена на рисунке 5.19.

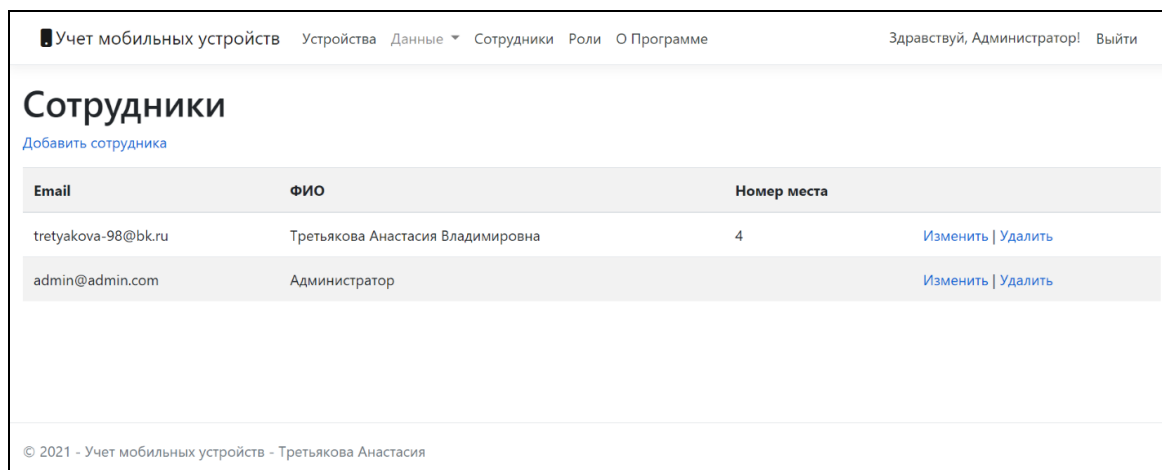


Рисунок 5.19 – Форма «Сотрудники»

Форма «Добавление сотрудника» представлена на рисунке 5.20.

Учет мобильных устройств Устройства Данные ▾ Сотрудники Роли О Программе Здравствуй, Администратор! Выйти

Сотрудник

Email

ФИО

Номер места

Пароль

Роли

☐ Администратор
☐ Пользователь

[Создать](#)

[Вернуться к списку](#)

© 2021 - Учет мобильных устройств - Третьякова Анастасия

Рисунок 5.20 – Форма «Добавление сотрудника»

После сохранения введенных данных, они отобразятся на форме «Сотрудники», при нажатии на кнопку «Вернуться к списку» данные о сотруднике не будут сохранены.

Все данные доступны для удаления администратору, для этого необходимо нажать кнопку «Удалить».

ОБОСНОВАНИЕ

6.1 Описание функций, назначения и потенциальных пользователей программного средства

Целью данного дипломного проекта является создание веб-приложения для ведения учета мобильных устройств департамента Комплексных решений ЗАО «Кьюликс Системс».

Разрабатываемое веб-приложение предназначено для применения командой тестирования департамента Комплексных решений ЗАО «Кьюликс Системс». Данное веб-приложение позволяет отследить, какой сотрудник использует конкретное устройство в данный момент и где устройство находится (дома или же в офисе). Так же данное веб-приложение позволяет создавать, редактировать и удалять информацию об устройствах департамента и о сотрудниках.

Программное средство разрабатывается для собственных нужд организации.

6.2 Расчет затрат на разработку программного средства

Затраты на разработку программного средства включают в себя следующие статьи:

- затраты на основную заработную плату разработчиков;
- затраты на дополнительную заработную плату разработчиков;
- отчисления на социальные нужды;
- прочие затраты (амортизационные отчисления, расходы на электроэнергию, командировочные расходы, арендная плата за офисные помещения и оборудование, расходы на управление и реализацию и т.п.).

6.2.1 Расчет затрат на основную заработную плату команды разработчиков

Затраты на основную заработную плату определяются составом команды, которая занимается разработкой программного средств, месячным окладом специалистов и трудоемкостью процесса разработки и рассчитываются по формуле:

$$Z_0 = K_{\text{пр}} \cdot \sum_{i=1}^n T_{\text{ч}i} \cdot t_i, \quad (6.1)$$

где n – количество исполнителей, занятых разработкой конкретного ПС;

$K_{пр}$ – коэффициент премий;

T_{ci} – часовая заработная плата i -го исполнителя, руб.;

t_i – трудоемкость работ, выполняемых i -м исполнителем, ч.

В разработке веб-приложения будет участвовать 3 исполнителя.

Данные по заработной плате команды разработчиков предоставлены ЗАО «Кьюликс Системс» на 29.10.2021.

Расчет затрат на основную заработную плату команды разработчиков представлено в таблице 6.1.

Таблица 6.1 – Расчет затрат на основную заработную плату команды разработчиков

Участник команды	Вид выполняемой работы	Месячная заработная плата, руб.	Часовая заработная плата, руб.	Трудоемкость работ, ч.	Зарплата по тарифу, руб.
Инженер-программист	Разработка ПС	1780	10,60	248	2628,80
Бизнес-аналитик	Анализ требований	1396	8,30	80	664
Тестирующий	Тестирование ПС	1480	8,80	88	774,4
Премия (90% от основной заработной платы)					2365,92
Итого затраты на основную заработную плату разработчика					4994,72

6.2.2 Расчет затрат на дополнительную заработную плату

Затраты на дополнительную заработную плату разработчика включает выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (6.2)$$

где $З_о$ – затраты на основную заработную плату, руб.;

$Н_д$ – норматив дополнительной заработной платы (18%).

Затраты на дополнительную заработную плату составят:

$$З_д = \frac{4994,72 \cdot 18\%}{100} = 899,05 \text{ руб.}$$

6.2.3 Расчет отчислений на социальные нужды

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

$$P_{\text{соц}} = \frac{(З_о + З_д) \cdot H_{\text{соц}}}{100}, \quad (6.3)$$

где $H_{\text{соц}}$ – норматив отчислений на социальные нужды (34,6%).

Отчисления на социальные нужды составят:

$$P_{\text{соц}} = \frac{(4994,72 + 899,05) \cdot 34,6}{100} = 2039,24 \text{ руб.}$$

6.2.4 Расчет прочих затрат

Прочие затраты включают затраты, связанные с разработкой конкретного программного средства напрямую, а также связанные с функционированием организации-разработчика в целом. Расчет прочих затрат выполняется в процентах от затрат на основную заработную плату команды разработчиков с учетом премии по формуле:

$$З_{\text{пз}} = \frac{З_о \cdot H_{\text{пз}}}{100}, \quad (6.4)$$

где $H_{\text{пз}}$ – норматив прочих затрат (145%).

Рассчитаем сумму прочих затрат:

$$З_{\text{пз}} = \frac{4994,72 \cdot 145}{100} = 7242,34 \text{ руб.}$$

Полная сумма затрат на разработку программного средства находится путем суммирования всех рассчитанных статей затрат. Расчет приведен в таблице 6.2.

Таблица 6.2 – Затраты на разработку программного средства

Статья затрат	Сумма, руб.
Основная заработная плата разработчика	4994,72
Дополнительная заработная плата разработчика	899,05
Отчисления на социальные нужды	2039,24
Прочие затраты	7242,34
Общая сумма затрат на разработку	15175,35

Общая сумма затрат на разработку составила 15175,35 руб.

6.3 Оценка экономического эффекта от разработки и применения программного средства для собственных нужд организации

Расчет экономического эффекта в результате применения программного средства, разработанного для собственных нужд организации, рассчитывается по формуле:

$$\Delta\P_{\text{ч}} = (\mathcal{E}_{\text{з}} - \mathcal{Z}_{\text{р}} - \Delta\mathcal{Z}_{\text{тек}}) \cdot (1 - H_{\text{п}}), \quad (6.5)$$

где $\mathcal{Z}_{\text{р}}$ – затраты на разработку программного средства, руб.;

$\mathcal{E}_{\text{з}}$ – экономия текущих затрат на ремонте мобильных устройств и уменьшении издержек на тестирование рабочих проектов сотрудниками, за счет возможности использовать мобильные устройства вне офиса, полученная в результате применения ПС, руб.;

$\Delta\mathcal{Z}_{\text{тек}}$ – прирост текущих затрат на приобретение новых мобильных устройств, связанных с использованием ПС, руб.;

$H_{\text{п}}$ – ставка налога на прибыль, в соответствии с действующим законодательством (18%).

Экономия текущих затрат ($\mathcal{E}_{\text{з}}$) по данным организации на 23.11.2021 составляет 29378 руб.

Прирост текущих затрат ($\Delta\mathcal{Z}_{\text{тек}}$) по данным организации на 23.11.2021 составляет 3275 руб.

Рассчитаем экономический эффект:

$$\Delta\P_{\text{ч}} = (29378 - 15175,35 - 3275) \cdot (1 - 0,18) = 8960,67 \text{ руб.}$$

Также необходимо рассчитать рентабельность затрат на разработку ПС по формуле:

$$Y_{\text{р}} = \frac{\Delta\P_{\text{ч}}}{\mathcal{Z}_{\text{р}}} \cdot 100\% \quad (6.6)$$

Рассчитаем рентабельность затрат на разработку ПС:

$$Y_{\text{р}} = \frac{8960,67}{15175,35} \cdot 100\% = 59,05 \%$$

В данном случае экономический эффект при разработке программного средства «Веб-приложение для учета мобильных устройств ЗАО «Кьюликс Системс»» для собственных нужд составляет 8960,67 руб., а уровень его рентабельности составляет **59,05 %**. Затраты на разработку программного средства составили 15175,35 руб.

ЗАКЛЮЧЕНИЕ

В ходе дипломного проектирования проведен анализ литературы по теме дипломного проекта, изучены наиболее известные программные решения для учета различной техники на предприятии. В процессе работы были исследованы основные направления по ведению учета техники на предприятии.

Для каждого программного средства были выделены его достоинства и недостатки использования, которые учитывались при формировании требований к программному средству.

В ходе моделирования программного средства были разработаны функциональные требования и описана функциональность к программному средству. На основе требований была разработана информационная модель базы данных и схема вариантов использования.

Исходя из полученных на этапе моделирования данных, была спроектирована архитектура программного средства и разработаны структуры классов и спроектированы алгоритмы работы программного средства. На основании информационной модели была спроектирована модель базы данных.

После разработки программного средства было проведено функциональное тестирование, в ходе которого дефекты не были обнаружены.

Таким образом, задачи, поставленные в рамках индивидуального задания, были выполнены. Знания и опыт полученные в процессе прохождения дипломного проектирования будут полезны при дальнейшей работе по специальности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Моделирование на UML. Общие диаграммы [Электронный ресурс]. – Электронные данные. – Режим доступа: http://book.uml3.ru/sec_1_5 – Дата доступа: 18.11.2021
- [2] Проектирование реляционной базы данных [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://itteach.ru/bazi-dannich/proektirovanie-relyatsionnoy-bazi-dannich> – Дата доступа: 21.11.2021
- [3] Введение в ASP.NET Core [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0> – Дата доступа: 24.11.2021
- [4] Бураков, П.В. Введение в системы баз данных: учебное пособие / П.В. Бураков, В.Ю. Петров – СПб: СПбГУ ИТМО, 2010. – 128 с.
- [5] Горовой, В. Г. Экономическое обоснование проекта по разработке программного обеспечения : метод. пособие / В. Г. Горовой, А. В. Грицай, В.А. Пархименко. – Минск : БГУИР, 2018. – 12 с.
- [6] Виды учета [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://ta-aspect.by/project-materials/hozyajstvennyj-uchet> Дата доступа: 05.11.2021
- [7] Характеристика полной оперативной информации [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://infopedia.su/18x14bba.html> Дата доступа: 05.11.2021
- [8] ПО «Hardware Inspector» [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://www.hwinspector.com/> Дата доступа: 08.11.2021
- [9] ПО «Инвентаризация сети и учета и компьютеров» [Электронный ресурс]. – Электронный ресурс. – Режим доступа: https://studbooks.net/2226090/informatika/programmnyy_produkt_inventarizatsiya_seti_ucheta_kompyuterov Дата доступа: 09.11.2021
- [10] Диаграмма вариантов использования [Электронный ресурс]. – Электронный ресурс. – Режим доступа: http://book.uml3.ru/sec_1_5 Дата доступа: 11.11.2021
- [11] Общие сведения об ASP.NET MVC [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview> Дата доступа: 12.11.2021
- [12] Введение в язык C# и .NET Framework [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework> Дата доступа: 16.11.2021
- [13] Закрытое акционерное общество [Электронный ресурс]. – Электронный ресурс. – Режим доступа: <https://dic.academic.ru/dic.nsf/lower/14807> Дата доступа: 01.12.2021

ПРИЛОЖЕНИЕ А (обязательное)

Текст программы

```
@page
@model AccessDeniedModel
@{
    ViewData["Title"] = "Доступ закрыт";
}

<header class="text-center">
    <h1 class="text-danger">Доступ закрыт</h1>
    <p class="text-danger">У вас нет доступа к этому ресурсу.</p>
</header>

@page
@model ForgotPasswordModel
@{
    ViewData["Title"] = "Forgot your password?";
}

<h1>@ViewData["Title"]</h1>
<h4>Enter your email.</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <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>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

@page
@model LoginModel

@{
    ViewData["Title"] = "Вход в систему";
}

<h1>@ViewData["Title"]</h1>
<div class="row">
    <div class="col-md-4">
        <section>
            <form id="account" method="post">
                <div asp-validation-summary="All" class="text-danger"></div>
                <div class="form-group">
                    <label asp-for="Input.Email">Логин</label>
                    <input asp-for="Input.Email" class="form-control" />
                    <span asp-validation-for="Input.Email" class="text-danger"></span>
                </div>
            </form>
        </section>
    </div>
</div>
```

```

    </div>
    <div class="form-group">
        <label asp-for="Input.Password">Пароль</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">Войти</button>
    </div>
    <!-- <div class="form-group">
        <p>
            <a id="forgot-password" asp-page="./ForgotPassword">Забыл пароль?</a>
        </p>
        <p>
            <a id="resend-confirmation" asp-page="./ResendEmailConfirmation">Отправить
еще раз подтверждение на Email</a>
        </p>
    </div>-->
    </form>
</section>
</div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

@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" />
            </div>
        </form>
    </div>
    </div>

```

```

        <span asp-validation-for="Input.ConfirmPassword" class="text-
danger"></span>
    </div>
    <button type="submit" class="btn btn-primary">Register</button>
</form>
</div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

@page
@model ResetPasswordModel
@{
    ViewData["Title"] = "Reset password";
}

<h1>@ViewData["Title"]</h1>
<h4>Reset your password.</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input asp-for="Input.Code" type="hidden" />
            <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">Reset</button>
        </form>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

<environment include="Development">
    <script src="~/Identity/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/Identity/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment exclude="Development">
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/jquery.validate.min.js"
asp-fallback-src="~/Identity/lib/jquery-
validation/dist/jquery.validate.min.js"

```

```

        asp-fallback-test="window.jQuery && window.jQuery.validator"
        crossorigin="anonymous"
        integrity="sha384-
rZfj/ogBloos6wzLGpPkkOr/gpkBNLZ6b6yLy4o+ok+t/SAK1L5mvXlr00XNi1Hp">
    </script>
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.9/jquery.validate.
unobtrusive.min.js"
        asp-fallback-src=~/_/Identity/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"
        asp-fallback-test="window.jQuery && window.jQuery.validator &&
window.jQuery.validator.unobtrusive"
        crossorigin="anonymous"
        integrity="sha384-
ifv0TYDwxBHvAk2Z0n8R434FL1Rlv/Av18DXE43N/1rvHyOG4izKst0f2iSLdds">
    </script>
</environment>

```

```

@using Microsoft.AspNetCore.Identity
@using WebAppMobileRecord.Areas.Identity
@using WebAppMobileRecord.Areas.Identity.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using WebAppMobileRecord.Data

```

```

@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

```

```

using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using WebAppMobileRecord.Data;

```

```

[assembly:
HostingStartup(typeof(WebAppMobileRecord.Areas.Identity.IdentityHostingStartup))]
namespace WebAppMobileRecord.Areas.Identity
{
    public class IdentityHostingStartup : IHostingStartup
    {
        {
            public void Configure(IWebHostBuilder builder)
            {
                builder.ConfigureServices((context, services) => {
                });
            }
        }
    }
}

```

```

using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;

```



```

using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class AppUsersController : Controller
    {
        UserManager<AppUser> _userManager;
        RoleManager<IdentityRole> _roleManager;

        public AppUsersController(RoleManager<IdentityRole> roleManager,
            UserManager<AppUser> userManager)
        {
            _roleManager = roleManager;
            _userManager = userManager;
        }

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

        public async Task<IActionResult> Create()
        {
            ViewData["AllRoles"] = _roleManager.Roles.ToList();
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Create(AppUser model, string password,
            List<string> roles)
        {
            model.UserName = model.Email;
            var password1 = new PasswordHasher<AppUser>();
            var hashed = password1.HashPassword(model, password);
            model.PasswordHash = hashed;
            var result = await _userManager.CreateAsync(model);

            await _userManager.AddToRolesAsync(model, roles);
            if (result.Succeeded)
            {
                return RedirectToAction("Index");
            }
            else
            {
                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }

            return View(model);
        }

        public async Task<IActionResult> Edit(string id)
        {
            AppUser user = await _userManager.FindByIdAsync(id);
            if (user == null)
            {
                return NotFound();
            }

            ViewData["UserRoles"] = await _userManager.GetRolesAsync(user);
            ViewData["AllRoles"] = _roleManager.Roles.ToList();
            return View(user);
        }
    }
}

```

```

[HttpPost]
public async Task<IActionResult> Edit(AppUser model, List<string> roles)
{
    AppUser user = await _userManager.FindByIdAsync(model.Id);
    if (user != null)
    {
        user.Email = model.Email;
        user.UserName = model.Email;
        user.FullName = model.FullName;
        user.PlaceNumber = model.PlaceNumber;

        var result = await _userManager.UpdateAsync(user);
        var userRoles = await _userManager.GetRolesAsync(user);
        // получаем все роли
        var allRoles = _roleManager.Roles.ToList();
        // получаем список ролей, которые были добавлены
        var addedRoles = roles.Except(userRoles);
        // получаем роли, которые были удалены
        var removedRoles = userRoles.Except(roles);

        await _userManager.AddToRolesAsync(user, addedRoles);

        await _userManager.RemoveFromRolesAsync(user, removedRoles);
        if (result.Succeeded)
        {
            return RedirectToAction("Index");
        }
        else
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }
    }

    return View(model);
}

[HttpPost]
public async Task<ActionResult> Delete(string id)
{
    AppUser user = await _userManager.FindByIdAsync(id);
    if (user != null)
    {
        IdentityResult result = await _userManager.DeleteAsync(user);
    }

    return RedirectToAction("Index");
}
}

public class CreateUserViewModel
{
    public string Email { get; set; }
    public string Password { get; set; }
}

public class EditUserViewModel
{
    public string Id { get; set; }
    public string Email { get; set; }
}

```

```

        public string UserEmail { get; set; }
        public List<IdentityRole> AllRoles { get; set; }
        public IList<string> UserRoles { get; set; }

        public EditUserViewModel()
        {
            AllRoles = new List<IdentityRole>();
            UserRoles = new List<string>();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    public class AssignMobileIdentitiesController : Controller
    {
        private readonly ApplicationDbContext _context;

        public AssignMobileIdentitiesController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: AssignMobileIdentities
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.AssignMobileIdentities.Include(a =>
a.Identity).Include(a => a.Mobile);
            return View(await applicationDbContext.ToListAsync());
        }

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

            var assignMobileIdentity = await _context.AssignMobileIdentities
                .Include(a => a.Identity)
                .Include(a => a.Mobile)
                .FirstOrDefaultAsync(m => m.IdentityId == id);
            if (assignMobileIdentity == null)
            {
                return NotFound();
            }

            return View(assignMobileIdentity);
        }

        // GET: AssignMobileIdentities/Create

```

```

public IActionResult Create()
{
    ViewData["IdentityId"] = new SelectList(_context.Users, "Id", "Id");
    ViewData["MobileId"] = new SelectList(_context.Mobiles, "Id", "Id");
    return View();
}

// POST: AssignMobileIdentities/Create
// To protect from overposting attacks, enable the specific properties you want
to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("Id,AssignDate,UnAssignDate,IdentityId,MobileId")] AssignMobileIdentity
assignMobileIdentity)
{
    if (ModelState.IsValid)
    {
        _context.Add(assignMobileIdentity);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["IdentityId"] = new SelectList(_context.Users, "Id", "Id",
assignMobileIdentity.IdentityId);
    ViewData["MobileId"] = new SelectList(_context.Mobiles, "Id", "Id",
assignMobileIdentity.MobileId);
    return View(assignMobileIdentity);
}

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

    var assignMobileIdentity = await
_context.AssignMobileIdentities.FindAsync(id);
    if (assignMobileIdentity == null)
    {
        return NotFound();
    }
    ViewData["IdentityId"] = new SelectList(_context.Users, "Id", "Id",
assignMobileIdentity.IdentityId);
    ViewData["MobileId"] = new SelectList(_context.Mobiles, "Id", "Id",
assignMobileIdentity.MobileId);
    return View(assignMobileIdentity);
}

// POST: AssignMobileIdentities/Edit/5
// To protect from overposting attacks, enable the specific properties you want
to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("Id,AssignDate,UnAssignDate,IdentityId,MobileId")] AssignMobileIdentity
assignMobileIdentity)
{
    if (id != assignMobileIdentity.Id)
    {

```

```

        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(assignMobileIdentity);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!AssignMobileIdentityExists(assignMobileIdentity.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["IdentityId"] = new SelectList(_context.Users, "Id", "Id",
assignMobileIdentity.IdentityId);
    ViewData["MobileId"] = new SelectList(_context.Mobiles, "Id", "Id",
assignMobileIdentity.MobileId);
    return View(assignMobileIdentity);
}

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

    var assignMobileIdentity = await _context.AssignMobileIdentities
        .Include(a => a.Identity)
        .Include(a => a.Mobile)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (assignMobileIdentity == null)
    {
        return NotFound();
    }

    return View(assignMobileIdentity);
}

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

private bool AssignMobileIdentityExists(int id)

```

```

        {
            return _context.AssignMobileIdentities.Any(e => e.Id == id);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using WebAppMobileRecord.Data;
using WebAppMobileRecord.Models;

namespace WebAppMobileRecord.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        UserManager<AppUser> _userManager;
        private readonly ApplicationDbContext _context;
        RoleManager<IdentityRole> _roleManager;

        public HomeController(ILogger<HomeController> logger, RoleManager<IdentityRole>
roleManager, UserManager<AppUser> userManager, ApplicationDbContext context)
        {
            _logger = logger;
            _roleManager = roleManager;
            _userManager = userManager;
            _context = context;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        public IActionResult Description()
        {
            return View();
        }

        public async Task<IActionResult> CreateData()
        {
            await Initialize();

            return Ok();
        }

        public async Task Initialize()

```

```

{
    string[] roles = new string[] { "Администратор", "Пользователь" };

    foreach (string role in roles)
    {
        if (!_context.Roles.Any(r => r.Name == role))
        {
            await _roleManager.CreateAsync(new IdentityRole(role));
        }
    }

    var user = new AppUser
    {
        Email = "admin@admin.com",
        UserName = "admin@admin.com",
        FullName = "Администратор",
        NormalizedUserName = "ADMIN",
        EmailConfirmed = true,
        PhoneNumberConfirmed = true,
        SecurityStamp = Guid.NewGuid().ToString("D")
    };

    if (!_context.Users.Any(u => u.UserName == user.UserName))
    {
        var password = new PasswordHasher<AppUser>();
        var hashed = password.HashPassword(user, "1Admin!");
        user.PasswordHash = hashed;

        var result = await _userManager.CreateAsync(user);

        await _userManager.AddToRolesAsync(user, roles);
    }

    await _context.SaveChangesAsync();
}

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

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

```

```

using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize]
    public class MobilesController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<AppUser> _userManager;

        public MobilesController(ApplicationDbContext context, UserManager<AppUser>
userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        // GET: Mobiles
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.Mobiles
                .Include(m => m.AssignMobileIdentities)
                .ThenInclude(x => x.Identity).Include(m => m.MobileStatus).Include(m =>
m.MobileType)
                .Include(m => m.OSVersion).ThenInclude(m => m.OSType).Include(m =>
m.Vendor);
            return View(await applicationDbContext.ToListAsync());
        }

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

            var mobile = await _context.Mobiles
                .Include(m => m.MobileStatus)
                .Include(m => m.MobileType)
                .Include(m => m.OSVersion).ThenInclude(x => x.OSType)
                .Include(m => m.Vendor)
                .Include(m => m.AssignMobileIdentities)
                .ThenInclude(x => x.Identity)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (mobile == null)
            {
                return NotFound();
            }

            return View(mobile);
        }

        // GET: Mobiles/Create
        public IActionResult Create()
        {
            ViewData["MobileStatusId"] = new SelectList(_context.MobileStatuses, "Id",
"StatusName");
            ViewData["MobileTypeId"] = new SelectList(_context.MobileTypes, "Id",
"TypeName");
            ViewData["OSVersionId"] = new SelectList(_context.OSVersions, "Id",
"Version");
            ViewData["VendorId"] = new SelectList(_context.Vendors, "Id", "VendorName");
        }
    }
}

```



```

        return View();
    }

    // POST: Mobiles/Create
    // To protect from overposting attacks, enable the specific properties you want
    // to bind to, for
    // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create(
[Bind("Id,Number,AddedDate,DeactivatedDate,OSVersionId,MobileStatusId,MobileTypeId,Vendor
Id,Comment")]
        Mobile mobile)
    {
        if (ModelState.IsValid)
        {
            _context.Add(mobile);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        ViewData["MobileStatusId"] =
            new SelectList(_context.MobileStatuses, "Id", "StatusName",
mobile.MobileStatusId);
        ViewData["MobileTypeId"] = new SelectList(_context.MobileTypes, "Id",
"TypeName", mobile.MobileTypeId);
        ViewData["OSVersionId"] = new SelectList(_context.OSVersions, "Id",
"Version", mobile.OSVersionId);
        ViewData["VendorId"] = new SelectList(_context.Vendors, "Id", "VendorName",
mobile.VendorId);
        return View(mobile);
    }

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

        var mobile = await _context.Mobiles.Include(m => m.MobileStatus)
            .Include(m => m.MobileType)
            .Include(m => m.OSVersion).ThenInclude(x => x.OSType)
            .Include(m => m.Vendor)
            .Include(m => m.AssignMobileIdentities)
            .ThenInclude(x => x.Identity).FirstOrDefaultAsync(x => x.Id == id);
        if (mobile == null)
        {
            return NotFound();
        }

        ViewData["MobileStatusId"] =
            new SelectList(_context.MobileStatuses, "Id", "StatusName",
mobile.MobileStatusId);
        ViewData["MobileTypeId"] = new SelectList(_context.MobileTypes, "Id",
"TypeName", mobile.MobileTypeId);
        ViewData["OSVersionId"] = new SelectList(_context.OSVersions, "Id",
"Version", mobile.OSVersionId);
        ViewData["VendorId"] = new SelectList(_context.Vendors, "Id", "VendorName",
mobile.VendorId);
    }

```

```

        ViewData["Users"] = new SelectList(_context.Users, "Id", "Email",
mobile.AssignMobileIdentities?.FirstOrDefault()?.IdentityId);
        return View(mobile);
    }

    // POST: Mobiles/Edit/5
    // To protect from overposting attacks, enable the specific properties you want
to bind to, for
    // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id,

[Bind("Id,Number,AddedDate,DeactivatedDate,OSVersionId,MobileStatusId,MobileTypeId,Vendor
Id,Comment")]
        Mobile mobile,
        [Bind("UserId")]
        string userId)
    {
        if (id != mobile.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(mobile);
                if (string.IsNullOrEmpty(userId))
                {
                    userId = _userManager.Users.FirstOrDefault(x => x.Email ==
User.Identity.Name)?.Id;
                }
                var mobileStatus = await
_context.MobileStatuses.FirstOrDefaultAsync(x=>x.Id == mobile.MobileStatusId);
                if (mobileStatus.StatusName == "Взят")
                {
                    var assignments = _context.AssignMobileIdentities.Where(x =>
x.MobileId == id && x.UnAssignDate == null);
                    foreach (var assignment in assignments)
                    {
                        assignment.UnAssignDate = DateTime.Now;
                    }

                    var newAssignment = new AssignMobileIdentity()
                    {
                        AssignDate = DateTime.Now,
                        IdentityId = userId,
                        MobileId = id,
                    };

                    _context.AssignMobileIdentities.Add(newAssignment);
                }
                else
                {
                    var assignments = _context.AssignMobileIdentities.Where(x =>
x.MobileId == id && x.UnAssignDate == null);
                    foreach (var assignment in assignments)
                    {
                        assignment.UnAssignDate = DateTime.Now;
                    }
                }
            }
        }
    }

```

```

        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!MobileExists(mobile.Id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return RedirectToAction(nameof(Index));
}

ViewData["MobileStatusId"] =
    new SelectList(_context.MobileStatuses, "Id", "StatusName",
mobile.MobileStatusId);
ViewData["MobileTypeId"] = new SelectList(_context.MobileTypes, "Id",
"TypeName", mobile.MobileTypeId);
ViewData["OSVersionId"] = new SelectList(_context.OSVersions, "Id",
"Version", mobile.OSVersionId);
ViewData["VendorId"] = new SelectList(_context.Vendors, "Id", "VendorName",
mobile.VendorId);
return View(mobile);
}

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

    var mobile = await _context.Mobiles
        .Include(m => m.MobileStatus)
        .Include(m => m.MobileType)
        .Include(m => m.OSVersion).ThenInclude(x => x.OSType)
        .Include(m => m.Vendor)
        .Include(m => m.AssignMobileIdentities)
        .ThenInclude(x => x.Identity)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (mobile == null)
    {
        return NotFound();
    }

    return View(mobile);
}

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

```

```

    }

    private bool MobileExists(int id)
    {
        return _context.Mobiles.Any(e => e.Id == id);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class MobileStatusController : Controller
    {
        private readonly ApplicationDbContext _context;

        public MobileStatusController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: MobileStatus
        public async Task<IActionResult> Index()
        {
            return View(await _context.MobileStatuses.ToListAsync());
        }

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

            var mobileStatus = await _context.MobileStatuses
                .FirstOrDefaultAsync(m => m.Id == id);
            if (mobileStatus == null)
            {
                return NotFound();
            }

            return View(mobileStatus);
        }

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

```

```

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

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

            var mobileStatus = await _context.MobileStatuses.FindAsync(id);
            if (mobileStatus == null)
            {
                return NotFound();
            }
            return View(mobileStatus);
        }

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

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(mobileStatus);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!MobileStatusExists(mobileStatus.Id))
                    {
                        return NotFound();
                    }
                    else
                    {

```

```

        throw;
    }
    }
    return RedirectToAction(nameof(Index));
}
return View(mobileStatus);
}

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

    var mobileStatus = await _context.MobileStatuses
        .FirstOrDefaultAsync(m => m.Id == id);
    if (mobileStatus == null)
    {
        return NotFound();
    }

    return View(mobileStatus);
}

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

private bool MobileStatusExists(int id)
{
    return _context.MobileStatuses.Any(e => e.Id == id);
}
}

}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class MobileTypesController : Controller
    {
        private readonly ApplicationDbContext _context;
    }
}

```

```

public MobileTypesController(ApplicationDbContext context)
{
    _context = context;
}

// GET: MobileTypes
public async Task<IActionResult> Index()
{
    return View(await _context.MobileTypes.ToListAsync());
}

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

    var mobileType = await _context.MobileTypes
        .FirstOrDefaultAsync(m => m.Id == id);
    if (mobileType == null)
    {
        return NotFound();
    }

    return View(mobileType);
}

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

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

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

    var mobileType = await _context.MobileTypes.FindAsync(id);
    if (mobileType == null)

```

```

        {
            return NotFound();
        }
        return View(mobileType);
    }

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

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

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

        var mobileType = await _context.MobileTypes
            .FirstOrDefaultAsync(m => m.Id == id);
        if (mobileType == null)
        {
            return NotFound();
        }

        return View(mobileType);
    }

    // POST: MobileTypes/Delete/5
    [HttpPost, ActionName("Delete")]

```



```

[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var mobileType = await _context.MobileTypes.FindAsync(id);
    _context.MobileTypes.Remove(mobileType);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool MobileTypeExists(int id)
{
    return _context.MobileTypes.Any(e => e.Id == id);
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class OSTypesController : Controller
    {
        private readonly ApplicationDbContext _context;

        public OSTypesController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: OSTypes
        public async Task<IActionResult> Index()
        {
            return View(await _context.OSTypes.ToListAsync());
        }

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

            var oSType = await _context.OSTypes
                .FirstOrDefaultAsync(m => m.Id == id);
            if (oSType == null)
            {
                return NotFound();
            }

            return View(oSType);
        }
    }
}

```

```

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

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

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

    var oType = await _context.OTypes.FindAsync(id);
    if (oType == null)
    {
        return NotFound();
    }
    return View(oType);
}

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

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(oType);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {

```

```

        if (!OSTypeExists(oSType.Id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return RedirectToAction(nameof(Index));
}
return View(oSType);
}

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

    var oSType = await _context.OSTypes
        .FirstOrDefaultAsync(m => m.Id == id);
    if (oSType == null)
    {
        return NotFound();
    }

    return View(oSType);
}

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

private bool OSTypeExists(int id)
{
    return _context.OSTypes.Any(e => e.Id == id);
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

```

```

namespace WebAppMobileRecord.Controllers
{

```

```

[Authorize(Roles = "Администратор")]
public class OSVersionsController : Controller
{
    private readonly ApplicationDbContext _context;

    public OSVersionsController(ApplicationDbContext context)
    {
        _context = context;
    }

    // GET: OSVersions
    public async Task<IActionResult> Index()
    {
        var applicationDbContext = _context.OSVersions.Include(o => o.OSType);
        return View(await applicationDbContext.ToListAsync());
    }

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

        var oSVersion = await _context.OSVersions
            .Include(o => o.OSType)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (oSVersion == null)
        {
            return NotFound();
        }

        return View(oSVersion);
    }

    // GET: OSVersions/Create
    public IActionResult Create()
    {
        ViewData["OSTypeId"] = new SelectList(_context.OSTypes, "Id", "OSTypeName");
        return View();
    }

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

```

```

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

    var oSVersion = await _context.OSVersions.FindAsync(id);
    if (oSVersion == null)
    {
        return NotFound();
    }
    ViewData["OSTypeId"] = new SelectList(_context.OSTypes, "Id", "OSTypeName",
oSVersion.OSTypeId);
    return View(oSVersion);
}

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

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(oSVersion);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!OSVersionExists(oSVersion.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["OSTypeId"] = new SelectList(_context.OSTypes, "Id", "OSTypeName",
oSVersion.OSTypeId);
    return View(oSVersion);
}

// GET: OSVersions/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {

```

```

        return NotFound();
    }

    var oSVersion = await _context.OSVersions
        .Include(o => o.OSType)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (oSVersion == null)
    {
        return NotFound();
    }

    return View(oSVersion);
}

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

private bool OSVersionExists(int id)
{
    return _context.OSVersions.Any(e => e.Id == id);
}
}

using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class RolesController : Controller
    {
        RoleManager<IdentityRole> _roleManager;
        UserManager<AppUser> _userManager;
        public RolesController(RoleManager<IdentityRole> roleManager,
            UserManager<AppUser> userManager)
        {
            _roleManager = roleManager;
            _userManager = userManager;
        }
        public IActionResult Index() => View(_roleManager.Roles.ToList());

        public IActionResult Create() => View();
        [HttpPost]
        public async Task<IActionResult> Create(string name)
        {
            if (!string.IsNullOrEmpty(name))

```

```

        {
            IdentityResult result = await _roleManager.CreateAsync(new
IdentityRole(name));
            if (result.Succeeded)
            {
                return RedirectToAction("Index");
            }
            else
            {
                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }
        }
        return View(name);
    }

    [HttpGet, ActionName("Delete")]
    public async Task<IActionResult> Delete([FromQuery]string name)
    {
        IdentityRole role = await _roleManager.FindByNameAsync(name);
        if (role != null)
        {
            IdentityResult result = await _roleManager.DeleteAsync(role);
        }
        return RedirectToAction("Index");
    }

    public async Task<IActionResult> Edit(string userId)
    {
        // получаем пользователя
        AppUser user = await _userManager.FindByIdAsync(userId);
        if (user != null)
        {
            // получем список ролей пользователя
            var userRoles = await _userManager.GetRolesAsync(user);
            var allRoles = _roleManager.Roles.ToList();
            ChangeRoleViewModel model = new ChangeRoleViewModel
            {
                UserId = user.Id,
                UserEmail = user.Email,
                UserRoles = userRoles,
                AllRoles = allRoles
            };
            return View(model);
        }

        return NotFound();
    }

    [HttpPost]
    public async Task<IActionResult> Edit(string userId, List<string> roles)
    {
        // получаем пользователя
        AppUser user = await _userManager.FindByIdAsync(userId);
        if (user != null)
        {
            // получем список ролей пользователя
            var userRoles = await _userManager.GetRolesAsync(user);
            // получаем все роли
            var allRoles = _roleManager.Roles.ToList();
            // получаем список ролей, которые были добавлены

```

```

        var addedRoles = roles.Except(userRoles);
        // получаем роли, которые были удалены
        var removedRoles = userRoles.Except(roles);

        await _userManager.AddToRolesAsync(user, addedRoles);

        await _userManager.RemoveFromRolesAsync(user, removedRoles);

        return RedirectToAction("UserList");
    }

    return NotFound();
}
}
}
public class ChangeRoleViewModel
{
    public string UserId { get; set; }
    public string UserEmail { get; set; }
    public List<IdentityRole> AllRoles { get; set; }
    public IList<string> UserRoles { get; set; }
    public ChangeRoleViewModel()
    {
        AllRoles = new List<IdentityRole>();
        UserRoles = new List<string>();
    }
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Controllers
{
    [Authorize(Roles = "Администратор")]
    public class VendorsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public VendorsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Vendors
        public async Task<IActionResult> Index()
        {
            return View(await _context.Vendors.ToListAsync());
        }

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

```



```

    {
        return NotFound();
    }

    var vendor = await _context.Vendors
        .FirstOrDefaultAsync(m => m.Id == id);
    if (vendor == null)
    {
        return NotFound();
    }

    return View(vendor);
}

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

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

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

    var vendor = await _context.Vendors.FindAsync(id);
    if (vendor == null)
    {
        return NotFound();
    }
    return View(vendor);
}

// POST: Vendors/Edit/5
// To protect from overposting attacks, enable the specific properties you want
to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id, VendorName")] Vendor
vendor)
{
    if (id != vendor.Id)

```

```

    {
        return NotFound();
    }

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

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

    var vendor = await _context.Vendors
        .FirstOrDefaultAsync(m => m.Id == id);
    if (vendor == null)
    {
        return NotFound();
    }

    return View(vendor);
}

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

private bool VendorExists(int id)
{
    return _context.Vendors.Any(e => e.Id == id);
}
}
}

```

```

using System;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Migrations;

namespace WebAppMobileRecord.Data.Migrations
{
    public partial class CreateIdentitySchema : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "AspNetRoles",
                columns: table => new
                {
                    Id = table.Column<string>(nullable: false),
                    Name = table.Column<string>(maxLength: 256, nullable: true),
                    NormalizedName = table.Column<string>(maxLength: 256, nullable:
true),
                    ConcurrencyStamp = table.Column<string>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetRoles", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "AspNetUsers",
                columns: table => new
                {
                    Id = table.Column<string>(nullable: false),
                    UserName = table.Column<string>(maxLength: 256, nullable: true),
                    NormalizedUserName = table.Column<string>(maxLength: 256, nullable:
true),
                    Email = table.Column<string>(maxLength: 256, nullable: true),
                    NormalizedEmail = table.Column<string>(maxLength: 256, nullable:
true),
                    EmailConfirmed = table.Column<bool>(nullable: false),
                    PasswordHash = table.Column<string>(nullable: true),
                    SecurityStamp = table.Column<string>(nullable: true),
                    ConcurrencyStamp = table.Column<string>(nullable: true),
                    PhoneNumber = table.Column<string>(nullable: true),
                    PhoneNumberConfirmed = table.Column<bool>(nullable: false),
                    TwoFactorEnabled = table.Column<bool>(nullable: false),
                    LockoutEnd = table.Column<DateTimeOffset>(nullable: true),
                    LockoutEnabled = table.Column<bool>(nullable: false),
                    AccessFailedCount = table.Column<int>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUsers", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "AspNetRoleClaims",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn),
                    RoleId = table.Column<string>(nullable: false),
                    ClaimType = table.Column<string>(nullable: true),
                    ClaimValue = table.Column<string>(nullable: true)
                },

```

```

constraints: table =>
{
    table.PrimaryKey("PK_AspNetRoleClaims", x => x.Id);
    table.ForeignKey(
        name: "FK_AspNetRoleClaims_AspNetRoles_RoleId",
        column: x => x.RoleId,
        principalTable: "AspNetRoles",
        principalColumn: "Id",
        onDelete: ReferentialAction.Cascade);
});

migrationBuilder.CreateTable(
    name: "AspNetUserClaims",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn),
        UserId = table.Column<string>(nullable: false),
        ClaimType = table.Column<string>(nullable: true),
        ClaimValue = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_AspNetUserClaims", x => x.Id);
        table.ForeignKey(
            name: "FK_AspNetUserClaims_AspNetUsers_UserId",
            column: x => x.UserId,
            principalTable: "AspNetUsers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "AspNetUserLogins",
    columns: table => new
    {
        LoginProvider = table.Column<string>(maxLength: 128, nullable:
false),
        ProviderKey = table.Column<string>(maxLength: 128, nullable: false),
        ProviderDisplayName = table.Column<string>(nullable: true),
        UserId = table.Column<string>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_AspNetUserLogins", x => new { x.LoginProvider,
x.ProviderKey });
        table.ForeignKey(
            name: "FK_AspNetUserLogins_AspNetUsers_UserId",
            column: x => x.UserId,
            principalTable: "AspNetUsers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "AspNetUserRoles",
    columns: table => new
    {
        UserId = table.Column<string>(nullable: false),
        RoleId = table.Column<string>(nullable: false)
    },
    constraints: table =>

```

```

        {
            table.PrimaryKey("PK_AspNetUserRoles", x => new { x.UserId, x.RoleId
        });
        table.ForeignKey(
            name: "FK_AspNetUserRoles_AspNetRoles_RoleId",
            column: x => x.RoleId,
            principalTable: "AspNetRoles",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_AspNetUserRoles_AspNetUsers_UserId",
            column: x => x.UserId,
            principalTable: "AspNetUsers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "AspNetUserTokens",
    columns: table => new
    {
        UserId = table.Column<string>(nullable: false),
        LoginProvider = table.Column<string>(maxLength: 128, nullable:
false),
        Name = table.Column<string>(maxLength: 128, nullable: false),
        Value = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_AspNetUserTokens", x => new { x.UserId,
x.LoginProvider, x.Name });
        table.ForeignKey(
            name: "FK_AspNetUserTokens_AspNetUsers_UserId",
            column: x => x.UserId,
            principalTable: "AspNetUsers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateIndex(
    name: "IX_AspNetRoleClaims_RoleId",
    table: "AspNetRoleClaims",
    column: "RoleId");

migrationBuilder.CreateIndex(
    name: "RoleNameIndex",
    table: "AspNetRoles",
    column: "NormalizedName",
    unique: true,
    filter: "[NormalizedName] IS NOT NULL");

migrationBuilder.CreateIndex(
    name: "IX_AspNetUserClaims_UserId",
    table: "AspNetUserClaims",
    column: "UserId");

migrationBuilder.CreateIndex(
    name: "IX_AspNetUserLogins_UserId",
    table: "AspNetUserLogins",
    column: "UserId");

migrationBuilder.CreateIndex(
    name: "IX_AspNetUserRoles_RoleId",

```

```

        table: "AspNetUserRoles",
        column: "RoleId");

migrationBuilder.CreateIndex(
    name: "EmailIndex",
    table: "AspNetUsers",
    column: "NormalizedEmail");

migrationBuilder.CreateIndex(
    name: "UserNameIndex",
    table: "AspNetUsers",
    column: "NormalizedUserName",
    unique: true,
    filter: "[NormalizedUserName] IS NOT NULL");
}

protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "AspNetRoleClaims");

    migrationBuilder.DropTable(
        name: "AspNetUserClaims");

    migrationBuilder.DropTable(
        name: "AspNetUserLogins");

    migrationBuilder.DropTable(
        name: "AspNetUserRoles");

    migrationBuilder.DropTable(
        name: "AspNetUserTokens");

    migrationBuilder.DropTable(
        name: "AspNetRoles");

    migrationBuilder.DropTable(
        name: "AspNetUsers");
}
}
using System;
using Microsoft.EntityFrameworkCore.Migrations;

namespace WebAppMobileRecord.Data.Migrations
{
    public partial class Init : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<int>(
                name: "DepartmentId",
                table: "AspNetUsers",
                nullable: true);

            migrationBuilder.AddColumn<string>(
                name: "FullName",
                table: "AspNetUsers",
                nullable: true);

            migrationBuilder.AddColumn<string>(
                name: "PlaceNumber",
                table: "AspNetUsers",
                nullable: true);
        }
    }
}

```

```

migrationBuilder.CreateTable(
    name: "Departments",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Name = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Departments", x => x.Id);
    });

migrationBuilder.CreateTable(
    name: "MobileStatuses",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        StatusName = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_MobileStatuses", x => x.Id);
    });

migrationBuilder.CreateTable(
    name: "MobileTypes",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        TypeName = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_MobileTypes", x => x.Id);
    });

migrationBuilder.CreateTable(
    name: "OSTypes",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        OSTypeName = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_OSTypes", x => x.Id);
    });

migrationBuilder.CreateTable(
    name: "Vendors",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        VendorName = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {

```

```

        table.PrimaryKey("PK_Vendors", x => x.Id);
    });

migrationBuilder.CreateTable(
    name: "OSVersions",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Version = table.Column<string>(nullable: true),
        OSTypeId = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_OSVersions", x => x.Id);
        table.ForeignKey(
            name: "FK_OSVersions_OSTypes_OSTypeId",
            column: x => x.OSTypeId,
            principalTable: "OSTypes",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "Mobiles",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Number = table.Column<string>(nullable: true),
        AddedDate = table.Column<DateTime>(nullable: false),
        DeactivatedDate = table.Column<DateTime>(nullable: true),
        OSVersionId = table.Column<int>(nullable: false),
        MobileStatusId = table.Column<int>(nullable: false),
        MobileTypeId = table.Column<int>(nullable: false),
        VendorId = table.Column<int>(nullable: false),
        Comment = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Mobiles", x => x.Id);
        table.ForeignKey(
            name: "FK_Mobiles_MobileStatuses_MobileStatusId",
            column: x => x.MobileStatusId,
            principalTable: "MobileStatuses",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Mobiles_MobileTypes_MobileTypeId",
            column: x => x.MobileTypeId,
            principalTable: "MobileTypes",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Mobiles_OSVersions_OSVersionId",
            column: x => x.OSVersionId,
            principalTable: "OSVersions",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Mobiles_Vendors_VendorId",
            column: x => x.VendorId,
            principalTable: "Vendors",

```



```

        principalColumn: "Id",
        onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "AssignMobileIdentities",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        AssignDate = table.Column<DateTime>(nullable: false),
        UnAssignDate = table.Column<DateTime>(nullable: true),
        IdentityId = table.Column<string>(nullable: true),
        MobileId = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_AssignMobileIdentities", x => x.Id);
        table.ForeignKey(
            name: "FK_AssignMobileIdentities_AspNetUsers_IdentityId",
            column: x => x.IdentityId,
            principalTable: "AspNetUsers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);
        table.ForeignKey(
            name: "FK_AssignMobileIdentities_Mobiles_MobileId",
            column: x => x.MobileId,
            principalTable: "Mobiles",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateIndex(
    name: "IX_AspNetUsers_DepartmentId",
    table: "AspNetUsers",
    column: "DepartmentId");

migrationBuilder.CreateIndex(
    name: "IX_AssignMobileIdentities_IdentityId",
    table: "AssignMobileIdentities",
    column: "IdentityId");

migrationBuilder.CreateIndex(
    name: "IX_AssignMobileIdentities_MobileId",
    table: "AssignMobileIdentities",
    column: "MobileId");

migrationBuilder.CreateIndex(
    name: "IX_Mobiles_MobileStatusId",
    table: "Mobiles",
    column: "MobileStatusId");

migrationBuilder.CreateIndex(
    name: "IX_Mobiles_MobileTypeId",
    table: "Mobiles",
    column: "MobileTypeId");

migrationBuilder.CreateIndex(
    name: "IX_Mobiles_OSVersionId",
    table: "Mobiles",
    column: "OSVersionId");

migrationBuilder.CreateIndex(

```

```

        name: "IX_Mobiles_VendorId",
        table: "Mobiles",
        column: "VendorId");

migrationBuilder.CreateIndex(
    name: "IX_OSVersions_OSTypeId",
    table: "OSVersions",
    column: "OSTypeId");

migrationBuilder.AddForeignKey(
    name: "FK_AspNetUsers_Departments_DepartmentId",
    table: "AspNetUsers",
    column: "DepartmentId",
    principalTable: "Departments",
    principalColumn: "Id",
    onDelete: ReferentialAction.Restrict);
}

protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropForeignKey(
        name: "FK_AspNetUsers_Departments_DepartmentId",
        table: "AspNetUsers");

    migrationBuilder.DropTable(
        name: "AssignMobileIdentities");

    migrationBuilder.DropTable(
        name: "Departments");

    migrationBuilder.DropTable(
        name: "Mobiles");

    migrationBuilder.DropTable(
        name: "MobileStatuses");

    migrationBuilder.DropTable(
        name: "MobileTypes");

    migrationBuilder.DropTable(
        name: "OSVersions");

    migrationBuilder.DropTable(
        name: "Vendors");

    migrationBuilder.DropTable(
        name: "OSTypes");

    migrationBuilder.DropIndex(
        name: "IX_AspNetUsers_DepartmentId",
        table: "AspNetUsers");

    migrationBuilder.DropColumn(
        name: "DepartmentId",
        table: "AspNetUsers");

    migrationBuilder.DropColumn(
        name: "FullName",
        table: "AspNetUsers");

    migrationBuilder.DropColumn(
        name: "PlaceNumber",
        table: "AspNetUsers");
}

```

```

    }
}
}
// <auto-generated />
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using WebAppMobileRecord.Data;

namespace WebAppMobileRecord.Data.Migrations
{
    [DbContext(typeof(ApplicationDbContext))]
    partial class ApplicationDbContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "3.1.21")
                .HasAnnotation("Relational:MaxIdentifierLength", 128)
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>
            {
                b.Property<string>("Id")
                    .HasColumnType("nvarchar(450)");

                b.Property<string>("ConcurrencyStamp")
                    .IsConcurrencyToken()
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("Name")
                    .HasColumnType("nvarchar(256)")
                    .HasMaxLength(256);

                b.Property<string>("NormalizedName")
                    .HasColumnType("nvarchar(256)")
                    .HasMaxLength(256);

                b.HasKey("Id");

                b.HasIndex("NormalizedName")
                    .IsUnique()
                    .HasName("RoleNameIndex")
                    .HasFilter("[NormalizedName] IS NOT NULL");

                b.ToTable("AspNetRoles");
            });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("int")
                    .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                b.Property<string>("ClaimType")
                    .HasColumnType("nvarchar(max)");
            });
        }
    }
}

```

```

        b.Property<string>("ClaimValue")
            .HasColumnType("nvarchar(max)");

        b.Property<string>("RoleId")
            .IsRequired()
            .HasColumnType("nvarchar(450)");

        b.HasKey("Id");

        b.HasIndex("RoleId");

        b.ToTable("AspNetRoleClaims");
    });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("ClaimType")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("ClaimValue")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("UserId")
        .IsRequired()
        .HasColumnType("nvarchar(450)");

    b.HasKey("Id");

    b.HasIndex("UserId");

    b.ToTable("AspNetUserClaims");
});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>
{
    b.Property<string>("LoginProvider")
        .HasColumnType("nvarchar(128)")
        .HasMaxLength(128);

    b.Property<string>("ProviderKey")
        .HasColumnType("nvarchar(128)")
        .HasMaxLength(128);

    b.Property<string>("ProviderDisplayName")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("UserId")
        .IsRequired()
        .HasColumnType("nvarchar(450)");

    b.HasKey("LoginProvider", "ProviderKey");

    b.HasIndex("UserId");

```

```

        b.ToTable("AspNetUserLogins");
    });

    modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>",
b =>
    {
        b.Property<string>("UserId")
            .HasColumnType("nvarchar(450)");

        b.Property<string>("RoleId")
            .HasColumnType("nvarchar(450)");

        b.HasKey("UserId", "RoleId");

        b.HasIndex("RoleId");

        b.ToTable("AspNetUserRoles");
    });

    modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>
    {
        b.Property<string>("UserId")
            .HasColumnType("nvarchar(450)");

        b.Property<string>("LoginProvider")
            .HasColumnType("nvarchar(128)")
            .HasMaxLength(128);

        b.Property<string>("Name")
            .HasColumnType("nvarchar(128)")
            .HasMaxLength(128);

        b.Property<string>("Value")
            .HasColumnType("nvarchar(max)");

        b.HasKey("UserId", "LoginProvider", "Name");

        b.ToTable("AspNetUserTokens");
    });

    modelBuilder.Entity("WebAppMobileRecord.Data.AppUser", b =>
    {
        b.Property<string>("Id")
            .HasColumnType("nvarchar(450)");

        b.Property<int>("AccessFailedCount")
            .HasColumnType("int");

        b.Property<string>("ConcurrencyStamp")
            .IsConcurrencyToken()
            .HasColumnType("nvarchar(max)");

        b.Property<int?>("DepartmentId")
            .HasColumnType("int");

        b.Property<string>("Email")
            .HasColumnType("nvarchar(256)")
            .HasMaxLength(256);

        b.Property<bool>("EmailConfirmed")
            .HasColumnType("bit");
    });

```

```

        b.Property<string>("FullName")
            .HasColumnType("nvarchar(max)");

        b.Property<bool>("LockoutEnabled")
            .HasColumnType("bit");

        b.Property<DateTimeOffset?>("LockoutEnd")
            .HasColumnType("datetimeoffset");

        b.Property<string>("NormalizedEmail")
            .HasColumnType("nvarchar(256)")
            .HasMaxLength(256);

        b.Property<string>("NormalizedUserName")
            .HasColumnType("nvarchar(256)")
            .HasMaxLength(256);

        b.Property<string>("PasswordHash")
            .HasColumnType("nvarchar(max)");

        b.Property<string>("PhoneNumber")
            .HasColumnType("nvarchar(max)");

        b.Property<bool>("PhoneNumberConfirmed")
            .HasColumnType("bit");

        b.Property<string>("PlaceNumber")
            .HasColumnType("nvarchar(max)");

        b.Property<string>("SecurityStamp")
            .HasColumnType("nvarchar(max)");

        b.Property<bool>("TwoFactorEnabled")
            .HasColumnType("bit");

        b.Property<string>("UserName")
            .HasColumnType("nvarchar(256)")
            .HasMaxLength(256);

        b.HasKey("Id");

        b.HasIndex("DepartmentId");

        b.HasIndex("NormalizedEmail")
            .HasName("EmailIndex");

        b.HasIndex("NormalizedUserName")
            .IsUnique()
            .HasName("UserNameIndex")
            .HasFilter("[NormalizedUserName] IS NOT NULL");

        b.ToTable("AspNetUsers");
    });

modelBuilder.Entity("WebAppMobileRecord.Data.AssignMobileIdentity", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<DateTime>("AssignDate")

```

```

        .HasColumnType("datetime2");

        b.Property<string>("IdentityId")
            .HasColumnType("nvarchar(450)");

        b.Property<int>("MobileId")
            .HasColumnType("int");

        b.Property<DateTime?>("UnAssignDate")
            .HasColumnType("datetime2");

        b.HasKey("Id");

        b.HasIndex("IdentityId");

        b.HasIndex("MobileId");

        b.ToTable("AssignMobileIdentities");
    });

modelBuilder.Entity("WebAppMobileRecord.Data.Department", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Name")
        .HasColumnType("nvarchar(max)");

    b.HasKey("Id");

    b.ToTable("Departments");
});

modelBuilder.Entity("WebAppMobileRecord.Data.Mobile", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<DateTime>("AddedDate")
        .HasColumnType("datetime2");

    b.Property<string>("Comment")
        .HasColumnType("nvarchar(max)");

    b.Property<DateTime?>("DeactivatedDate")
        .HasColumnType("datetime2");

    b.Property<int>("MobileStatusId")
        .HasColumnType("int");

    b.Property<int>("MobileTypeId")
        .HasColumnType("int");

    b.Property<string>("Number")
        .HasColumnType("nvarchar(max)");

    b.Property<int>("OSVersionId")

```

```

        .HasColumnType("int");

        b.Property<int>("VendorId")
            .HasColumnType("int");

        b.HasKey("Id");

        b.HasIndex("MobileStatusId");

        b.HasIndex("MobileTypeId");

        b.HasIndex("OSVersionId");

        b.HasIndex("VendorId");

        b.ToTable("Mobiles");
    });

modelBuilder.Entity("WebAppMobileRecord.Data.MobileStatus", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("StatusName")
        .HasColumnType("nvarchar(max)");

    b.HasKey("Id");

    b.ToTable("MobileStatuses");
});

modelBuilder.Entity("WebAppMobileRecord.Data.MobileType", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("TypeName")
        .HasColumnType("nvarchar(max)");

    b.HasKey("Id");

    b.ToTable("MobileTypes");
});

modelBuilder.Entity("WebAppMobileRecord.Data.OSType", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("OSTypeName")
        .HasColumnType("nvarchar(max)");

    b.HasKey("Id");

```



```

        b.ToTable("OSTypes");
    });

    modelBuilder.Entity("WebAppMobileRecord.Data.OSVersion", b =>
    {
        b.Property<int>("Id")
            .ValueGeneratedOnAdd()
            .HasColumnType("int")
            .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

        b.Property<int>("OSTypeId")
            .HasColumnType("int");

        b.Property<string>("Version")
            .HasColumnType("nvarchar(max)");

        b.HasKey("Id");

        b.HasIndex("OSTypeId");

        b.ToTable("OSVersions");
    });

    modelBuilder.Entity("WebAppMobileRecord.Data.Vendor", b =>
    {
        b.Property<int>("Id")
            .ValueGeneratedOnAdd()
            .HasColumnType("int")
            .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

        b.Property<string>("VendorName")
            .HasColumnType("nvarchar(max)");

        b.HasKey("Id");

        b.ToTable("Vendors");
    });

    modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>
    {
        b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)
            .WithMany()
            .HasForeignKey("RoleId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

    modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.AppUser", null)
            .WithMany()
            .HasForeignKey("UserId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

    modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>
    {

```

```

        b.HasOne("WebAppMobileRecord.Data.AppUser", null)
            .WithMany()
            .HasForeignKey("UserId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>",
b =>
    {
        b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)
            .WithMany()
            .HasForeignKey("RoleId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();

        b.HasOne("WebAppMobileRecord.Data.AppUser", null)
            .WithMany()
            .HasForeignKey("UserId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.AppUser", null)
            .WithMany()
            .HasForeignKey("UserId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

modelBuilder.Entity("WebAppMobileRecord.Data.AppUser", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.Department", null)
            .WithMany("AppUsers")
            .HasForeignKey("DepartmentId");
    });

modelBuilder.Entity("WebAppMobileRecord.Data.AssignMobileIdentity", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.AppUser", "Identity")
            .WithMany("AssignMobileIdentities")
            .HasForeignKey("IdentityId");

        b.HasOne("WebAppMobileRecord.Data.Mobile", "Mobile")
            .WithMany("AssignMobileIdentities")
            .HasForeignKey("MobileId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();
    });

modelBuilder.Entity("WebAppMobileRecord.Data.Mobile", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.MobileStatus", "MobileStatus")
            .WithMany("Mobiles")
            .HasForeignKey("MobileStatusId")
            .OnDelete(DeleteBehavior.Cascade)
            .IsRequired();

        b.HasOne("WebAppMobileRecord.Data.MobileType", "MobileType")
            .WithMany("Mobiles")

```

```

        .HasForeignKey("MobileTypeId")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

        b.HasOne("WebAppMobileRecord.Data.OSVersion", "OSVersion")
        .WithMany("Mobiles")
        .HasForeignKey("OSVersionId")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

        b.HasOne("WebAppMobileRecord.Data.Vendor", "Vendor")
        .WithMany("Mobiles")
        .HasForeignKey("VendorId")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
    });

    modelBuilder.Entity("WebAppMobileRecord.Data.OSVersion", b =>
    {
        b.HasOne("WebAppMobileRecord.Data.OSType", "OSType")
        .WithMany("OsVersions")
        .HasForeignKey("OSTypeId")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
    });
#pragma warning restore 612, 618
    }
}

using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace WebAppMobileRecord.Data
{
    public class ApplicationDbContext : IdentityDbContext<AppUser>
    {
        public DbSet<Mobile> Mobiles { get; set; }
        public DbSet<MobileStatus> MobileStatuses { get; set; }
        public DbSet<MobileType> MobileTypes { get; set; }
        public DbSet<OSType> OSTypes { get; set; }
        public DbSet<OSVersion> OSVersions { get; set; }
        public DbSet<Vendor> Vendors { get; set; }
        public DbSet<AssignMobileIdentity> AssignMobileIdentities { get; set; }
        public DbSet<Department> Departments { get; set; }
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}

#nullable enable
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Identity;

namespace WebAppMobileRecord.Data

```

```

{
    public class AppUser: IdentityUser
    {
        [DisplayName("ФИО")]
        public string? FullName { get; set; }

        [DisplayName("Номер места")]
        public string? PlaceNumber { get; set; }

        public List<AssignMobileIdentity> AssignMobileIdentities { get; set; }
    }
}

using Microsoft.AspNetCore.Identity;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class AssignMobileIdentity
    {
        [Key]
        public int Id { get; set; }

        [DisplayName("Дата назначения")]
        public DateTime AssignDate { get; set; }

        [DisplayName("Дата снятия")]
        public DateTime? UnAssignDate { get; set; }

        [DisplayName("Сотрудник")]
        public string IdentityId { get; set; }

        [ForeignKey("IdentityId")]
        [DisplayName("Сотрудник")]
        public AppUser Identity { get; set; }

        [DisplayName("Устройство")]
        public int MobileId { get; set; }

        [DisplayName("Устройство")]
        public Mobile Mobile { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class Department
    {

```

```

        public int Id { get; set; }

        public string Name { get; set; }

        public List<AppUser> AppUsers { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class Mobile
    {
        public int Id { get; set; }

        [DisplayName("Номер устройства")]
        public string Number { get; set; }

        [DisplayName("Дата добавления")]
        public DateTime AddedDate { get; set; }

        [DisplayName("Дата деактивации")]
        public DateTime? DeactivatedDate { get; set; }

        [DisplayName("Версия ОС")]
        public int OSVersionId { get; set; }

        [DisplayName("Версия ОС")]
        public OSVersion OSVersion { get; set; }

        [DisplayName("Статус")]
        public int MobileStatusId { get; set; }

        [DisplayName("Статус")]
        public MobileStatus MobileStatus { get; set; }

        [DisplayName("Тип устройства")]
        public int MobileTypeId { get; set; }

        [DisplayName("Тип устройства")]
        public MobileType MobileType { get; set; }

        [DisplayName("Производитель")]
        public int VendorId { get; set; }

        [DisplayName("Производитель")]
        public Vendor Vendor { get; set; }

        [DisplayName("Комментарий")]
        public string? Comment { get; set; }

        public List<AssignMobileIdentity> AssignMobileIdentities { get; set; }
    }
}

```

```

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class MobileStatus
    {
        public int Id { get; set; }

        [DisplayName("Статус")]
        public string StatusName { get; set; }

        public List<Mobile> Mobiles { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class MobileType
    {
        public int Id { get; set; }

        [DisplayName("Тип устройства")]
        public string TypeName { get; set; }

        public List<Mobile> Mobiles { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class OSType
    {
        public int Id { get; set; }

        [DisplayName("Тип ОС")]
        public string OSTypeName { get; set; }

        public List<OSVersion> OsVersions { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;

```

```

using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class OSVersion
    {
        public int Id { get; set; }

        [DisplayName("Версия")]
        public string Version { get; set; }

        [DisplayName("Тип ОС")]
        public int OSTypeId { get; set; }

        [DisplayName("Тип ОС")]
        public OSType OSType { get; set; }

        public List<Mobile> Mobiles { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;

namespace WebAppMobileRecord.Data
{
    public class Vendor
    {
        public int Id {get;set;}

        [DisplayName("Поставщи")]
        public string VendorName { get; set; }

        public List<Mobile> Mobiles { get; set; }
    }
}

using System;

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

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

@using Microsoft.AspNetCore.Identity
@model WebAppMobileRecord.Data.AppUser
@{
    ViewBag.Title = "Сотрудник";
}
<h4>Сотрудник</h4>
<hr />
<div class="row">

```

```

<div class="col-md-4">
    <form asp-action="Create" asp-controller="AppUsers">
        <div asp-validation-summary="All" class="text-danger"></div>
        <div class="form-group">
            <label asp-for="Email" class="control-label">Email</label>
            <input type="text" asp-for="Email" class="form-control" />
        </div>
        <div class="form-group">
            <label asp-for="FullName" class="control-label"></label>
            <input type="text" asp-for="FullName" class="form-control" />
        </div>
        <div class="form-group">
            <label asp-for="PlaceNumber" class="control-label"></label>
            <input type="text" asp-for="PlaceNumber" class="form-control" />
        </div>
        <div class="form-group">
            <label name="Password" class="control-label">Пароль</label>
            <input type="password" name="Password" class="form-control" />
        </div>

        <h2>Роли</h2>
        <div class="form-group">
            @foreach (IdentityRole role in ViewBag.AllRoles)
            {
                <input type="checkbox" name="roles" value="@role.Name" />@role.Name
            }
        </div>

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

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

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

@using Microsoft.AspNetCore.Identity

@model WebAppMobileRecord.Data.AppUser
@{
    ViewBag.Title = "Редактирование пользователя";
}
<h4>Сотрудник</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit" asp-controller="AppUsers">

            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <input type="hidden" asp-for="Id" />
            </div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>

```



```

        <input type="text" asp-for="Email" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="FullName" class="control-label"></label>
        <input type="text" asp-for="FullName" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="PlaceNumber" class="control-label"></label>
        <input type="text" asp-for="PlaceNumber" class="form-control" />
    </div>

    <h2>Роли</h2>
    <div class="form-group">
        @foreach (IdentityRole role in ViewBag.AllRoles)
        {
            <input type="checkbox" name="roles" value="@role.Name"
                @(ViewBag.UserRoles.Contains(role.Name) ?
"checked=\"checked\" : "" ) />@role.Name <br />
        }
    </div>
    <div class="form-group">
        <input type="submit" value="Сохранить" class="btn btn-outline-secondary"
/>
    </div>

    </form>
</div>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

@model IEnumerable<WebAppMobileRecord.Data.AppUser>
@{
    ViewBag.Title = "Список пользователей";
}

<h1>Сотрудники</h1>

<p>
    <a asp-action="Create">Добавить сотрудника</a>
</p>

<table class="table table-striped">
    <tr>
        <th>Email</th>
        <th>ФИО</th>
        <th>Номер места</th>
        <th></th>
    </tr>
    @foreach (var user in Model)
    {
<tr>
        <td>@user.Email</td>
        <td>@user.FullName</td>
        <td>@user.PlaceNumber</td>
        <td>
            <a asp-action="Edit" asp-route-id="@user.Id">Изменить</a> |
            <a asp-action="Delete" asp-route-id="@user.Id">Удалить</a>
        </td>
    </tr>
    }
}

```

```

</tr>
    }
</table>
@model WebAppMobileRecord.Data.AssignMobileIdentity

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

<h1>Create</h1>

<h4>AssignMobileIdentity</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="Id" class="control-label"></label>
                <input asp-for="Id" class="form-control" />
                <span asp-validation-for="Id" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="AssignDate" class="control-label"></label>
                <input asp-for="AssignDate" class="form-control" />
                <span asp-validation-for="AssignDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnAssignDate" class="control-label"></label>
                <input asp-for="UnAssignDate" class="form-control" />
                <span asp-validation-for="UnAssignDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="IdentityId" class="control-label"></label>
                <select asp-for="IdentityId" class="form-control" asp-
items="ViewBag.IdentityId"></select>
            </div>
            <div class="form-group">
                <label asp-for="MobileId" class="control-label"></label>
                <select asp-for="MobileId" class="form-control" asp-
items="ViewBag.MobileId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

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

@model WebAppMobileRecord.Data.AssignMobileIdentity

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

```

```

<h1>Delete</h1>

<h3>Вы уверены что хотите удалить?</h3>
<div>
    <h4>AssignMobileIdentity</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.AssignDate)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.AssignDate)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.UnAssignDate)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.UnAssignDate)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Identity)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Identity.Id)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Mobile)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Mobile.Id)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="IdentityId" />
        <input type="submit" value="Удалить" class="btn btn-primary" /> |
        <a asp-action="Index">Вернуться к списку</a>
    </form>
</div>

```

```

@model WebAppMobileRecord.Data.AssignMobileIdentity

```

```

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

```

```

<h1>Details</h1>

<div>
    <h4>AssignMobileIdentity</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>
    </dl>

```

```

</dd>
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.AssignDate)
</dt>
<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.AssignDate)
</dd>
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.UnAssignDate)
</dt>
<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.UnAssignDate)
</dd>
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Identity)
</dt>
<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Identity.Id)
</dd>
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Mobile)
</dt>
<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Mobile.Id)
</dd>
</dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.IdentityId">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>

@model WebAppMobileRecord.Data.AssignMobileIdentity

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

<h1>Edit</h1>

<h4>AssignMobileIdentity</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="AssignDate" class="control-label"></label>
                <input asp-for="AssignDate" class="form-control" />
                <span asp-validation-for="AssignDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnAssignDate" class="control-label"></label>
                <input asp-for="UnAssignDate" class="form-control" />
                <span asp-validation-for="UnAssignDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="IdentityId" class="control-label"></label>
                <select asp-for="IdentityId" class="form-control" asp-
items="ViewBag.IdentityId"></select>
                <span asp-validation-for="IdentityId" class="text-danger"></span>
            </div>
        </form>
    </div>
</div>

```

```

        <div class="form-group">
            <label asp-for="MobileId" class="control-label"></label>
            <select asp-for="MobileId" class="form-control" asp-
items="ViewBag.MobileId"></select>
            <span asp-validation-for="MobileId" class="text-danger"></span>
        </div>
        <div class="form-group">
            <input type="submit" value="Сохранить" class="btn btn-primary" />
        </div>
    </form>
</div>
</div>
</div>

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

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

@model IEnumerable<WebAppMobileRecord.Data.AssignMobileIdentity>

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

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.AssignDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.UnAssignDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Identity)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Mobile)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Id)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.AssignDate)
                </td>
            </tr>
        }
    </tbody>
</table>

```

```

        @Html.DisplayFor(modelItem => item.UnAssignDate)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Identity.Id)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Mobile.Id)
    </td>
    <td>
        <a asp-action="Edit" asp-route-id="@item.IdentityId">Редактировать</a> |
        <a asp-action="Details" asp-route-id="@item.IdentityId">Детальная
форма</a> |
        <a asp-action="Delete" asp-route-id="@item.IdentityId">Удалить</a>
    </td>
    </tr>
}
</tbody>
</table>

@{
    ViewData["Title"] = "0 Программе";
}

<div align="left">
    <h1 class="display-4">Назначение программного средства</h1>
    <p>Программное средство разработано для учета мобильных устройств и имеет следующие
функции:</p>
    <ul>
        <li>авторизация сотрудника;</li>
        <li>добавление, редактирование и удаление информации о сотрудниках (доступно
только администратору);</li>
        <li>добавление, редактирование и удаление информации о мобильных устройствах
(доступно только администратору);</li>
        <li>изменение статуса мобильного устройства;</li>
        <li>просмотр информации о мобильных устройствах;</li>
        <li>просмотр информации о сотрудниках (доступно только администратору);</li>
        <li>комментирование состояния мобильного устройства (необходимо для быстрого
информирования о состоянии мобильного устройства (сбой в работе, нежелательное обновление
ОС)).</li>
    </ul>
    <p>Программное средство разработано учащейся группы 881074 Третьяковой Анастасией
Владимировной.</p>
</div>

@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps
with ASP.NET Core</a>.</p>
</div>

@{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>

<p>Use this page to detail your site's privacy policy.</p>

@model WebAppMobileRecord.Data.Mobile
@{

```

```

        ViewData["Title"] = "Устройство";
    }

<h4>Устройство</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="Number" class="control-label"></label>
                <input asp-for="Number" class="form-control" />
                <span asp-validation-for="Number" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="AddedDate" class="control-label"></label>
                <input asp-for="AddedDate" class="form-control" />
                <span asp-validation-for="AddedDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DeactivatedDate" class="control-label"></label>
                <input asp-for="DeactivatedDate" class="form-control" />
                <span asp-validation-for="DeactivatedDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OSVersionId" class="control-label"></label>
                <select asp-for="OSVersionId" class="form-control" asp-
items="ViewBag.OSVersionId"></select>
            </div>
            <div class="form-group">
                <label asp-for="MobileStatusId" class="control-label"></label>
                <select asp-for="MobileStatusId" class="form-control" asp-
items="ViewBag.MobileStatusId"></select>
            </div>
            <div class="form-group">
                <label asp-for="MobileTypeId" class="control-label"></label>
                <select asp-for="MobileTypeId" class="form-control" asp-
items="ViewBag.MobileTypeId"></select>
            </div>
            <div class="form-group">
                <label asp-for="VendorId" class="control-label"></label>
                <select asp-for="VendorId" class="form-control" asp-
items="ViewBag.VendorId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

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

@model WebAppMobileRecord.Data.Mobile

@{
    ViewData["Title"] = "Устройство";

```

```

}

<h3>Вы уверены что хотите удалить?</h3>
<div>
  <h4>Устройство</h4>
  <hr />
  <dl class="row">
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.Number)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.Number)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.AddedDate)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.AddedDate)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.DeactivatedDate)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.DeactivatedDate)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.OSVersion)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.OSVersion.Id)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.MobileStatus)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.MobileStatus.Id)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.MobileType)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.MobileType.Id)
    </dd>
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.Vendor)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.Vendor.Id)
    </dd>
  </dl>

  <form asp-action="Delete">
    <input type="hidden" asp-for="Id" />
    <input type="submit" value="Delete" class="btn btn-danger" /> |
    <a asp-action="Index">Вернуться к списку</a>
  </form>
</div>

@model WebAppMobileRecord.Data.Mobile

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

```



```

<div>
  <h4>Устройство</h4>
  <hr />
  <dl class="row">
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Number)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Number)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.AddedDate)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.AddedDate)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.DeactivatedDate)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.DeactivatedDate)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.OSVersion)
    </dt>
    <dd class="col-sm-10">
      @{
        var osVersion =
        $"{@Model.OSVersion.OSType.OSTypeName}:{@Model.OSVersion.Version}";
        @Html.DisplayFor(modelItem => osVersion)
      }
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.MobileStatus)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.MobileStatus.StatusName)
    </dd>
    <dt class="col-sm-2">
      Email Сотрудника
    </dt>
    <dd class="col-sm-10">
      @{
        var result = Model.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.Email ?? "";
        @Html.DisplayFor(modelItem => result)
      }
    </dd>
    <dt class="col-sm-2">
      ФИО Сотрудника
    </dt>
    <dd class="col-sm-10">
      @{
        var resultFullName = Model.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.FullName ?? "";
        @Html.DisplayFor(modelItem => resultFullName)
      }
    </dd>
    <dt class="col-sm-2">
      № Места Сотрудника
    </dt>
  </dl>

```

```

<dd class="col-sm-10">
    @{
        var resultPlace = Model.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.PlaceNumber ?? "";
        @Html.DisplayFor(modelItem => resultPlace)
    }
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.MobileType)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.MobileType.TypeName)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Vendor)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Vendor.VendorName)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Comment)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Comment)
</dd>
</dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>
@model WebAppMobileRecord.Data.Mobile
@{
    ViewData["Title"] = "Устройство";
    var isAdmin = User.IsInRole("Администратор");
}

<h4>Устройство</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" />
            @if (isAdmin)
            {
                <div class="form-group">
                    <label asp-for="Number" class="control-label"></label>
                    <input asp-for="Number" class="form-control" />
                    <span asp-validation-for="Number" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="AddedDate" class="control-label"></label>
                    <input asp-for="AddedDate" class="form-control" type="date" />
                    <span asp-validation-for="AddedDate" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="DeactivatedDate" class="control-label"></label>
                    <input asp-for="DeactivatedDate" class="form-control" type="date" />
                    <span asp-validation-for="DeactivatedDate" class="text-
danger"></span>

```

```

        </div>
        <div class="form-group">
            <label asp-for="OSVersion" class="control-label"></label>
            <select asp-for="OSVersionId" class="form-control" asp-
items="ViewBag.OSVersionId"></select>
            <span asp-validation-for="OSVersionId" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="MobileStatus" class="control-label"></label>
            <select asp-for="MobileStatusId" class="form-control" asp-
items="ViewBag.MobileStatusId"></select>
            <span asp-validation-for="MobileStatusId" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label for="UserId" class="control-label">Сотрудник</label>
            <select id="UserId" name="UserId" class="form-control" asp-
items="ViewBag.Users"></select>
            <span id="UserId" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="MobileType" class="control-label"></label>
            <select asp-for="MobileTypeId" class="form-control" asp-
items="ViewBag.MobileTypeId"></select>
            <span asp-validation-for="MobileTypeId" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Vendor" class="control-label"></label>
            <select asp-for="VendorId" class="form-control" asp-
items="ViewBag.VendorId"></select>
            <span asp-validation-for="VendorId" class="text-danger"></span>
        </div>

        <div class="form-group">
            <label asp-for="Comment" class="control-label"></label>
            <input asp-for="Comment" class="form-control" />
            <span asp-validation-for="Comment" class="text-danger"></span>
        </div>
    }
    else
    {

        <input type="hidden" asp-for="Number" />
        <input type="hidden" asp-for="OSVersionId" />
        <input type="hidden" asp-for="MobileTypeId" />
        <input type="hidden" asp-for="VendorId" />
        <div class="form-group">
            <label asp-for="MobileStatus" class="control-label"></label>
            <select asp-for="MobileStatusId" class="form-control" asp-
items="ViewBag.MobileStatusId"></select>
            <span asp-validation-for="MobileStatusId" class="text-danger"></span>
        </div>

        <div class="form-group">
            <label asp-for="Comment" class="control-label"></label>
            <input asp-for="Comment" class="form-control" />
            <span asp-validation-for="Comment" class="text-danger"></span>
        </div>
    }
    <div class="form-group">
        <input type="submit" value="Сохранить" class="btn btn-primary" />
    </div>
</form>
</div>

```

```

</div>

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.Mobile>

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

<h1>Устройства</h1>

@if (User.IsInRole("Администратор"))
{
    <p>
        <a asp-action="Create">Добавить новое устройство</a>
    </p>
}
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Number)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.OSVersion)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.MobileStatus)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.MobileType)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Vendor)
            </th>
            <th>
                Email Сотрудника
            </th>
            <th>
                ФИО Сотрудника
            </th>
            <th>
                № Места Сотрудника
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Number)
                </td>
                <td>
                    @{

```

```

        var OSVersion =
${item.OSVersion.OSType.OSTypeName}:{item.OSVersion.Version}";
        @Html.DisplayFor(modelItem => OSVersion)
    }
</td>
<td>
        @Html.DisplayFor(modelItem => item.MobileStatus.StatusName)
    </td>
<td>
        @Html.DisplayFor(modelItem => item.MobileType.TypeName)
    </td>
<td>
        @Html.DisplayFor(modelItem => item.Vendor.VendorName)
    </td>
<td>
        @{
            var result = item.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.Email ?? "";
            @Html.DisplayFor(modelItem => result)
        }
    </td>
<td>
        @{
            var resultFullName = item.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.FullName ?? "";
            @Html.DisplayFor(modelItem => resultFullName)
        }
    </td>
<td>
        @{
            var resultPlace = item.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.PlaceNumber ?? "";
            @Html.DisplayFor(modelItem => resultPlace)
        }
    </td>
    @if (User.IsInRole("Администратор"))
    {
        <td>
            <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
            <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
            <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
        </td>
    }
    else
    {
        <td>
            <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a>
            @{
                var userEmail = item.AssignMobileIdentities?.FirstOrDefault(a =>
a.UnAssignDate == null)?.Identity?.Email;
                var isEnabledChangeStatus = string.IsNullOrEmpty(userEmail) ||
User.Identity.Name == userEmail;
            }
            @if (isEnabledChangeStatus)
            {
                <a asp-action="Edit" asp-route-id="@item.Id">Изменить статус</a>
            }
        </td>
    }
</tr>
}
</tbody>

```

```

</table>@model WebAppMobileRecord.Data.MobileStatus

@{
    ViewData["Title"] = "Статус устройства";
}

<h4>Статус устройства</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="StatusName" class="control-label"></label>
                <input asp-for="StatusName" class="form-control" />
                <span asp-validation-for="StatusName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model WebAppMobileRecord.Data.MobileStatus

@{
    ViewData["Title"] = "Статус устройства";
}

<h3>Вы уверены что хотите удалить?</h3>
<div>
    <h4>Статус устройства</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.StatusName)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.StatusName)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Удалить" class="btn btn-primary" /> |
        <a asp-action="Index">Вернуться к списку</a>
    </form>
</div>
@model WebAppMobileRecord.Data.MobileStatus

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

<div>

```

```

<h4>Статус устройства</h4>
<hr />
<dl class="row">
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.StatusName)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.StatusName)
    </dd>
</dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>
@model WebAppMobileRecord.Data.MobileStatus

@{
    ViewData["Title"] = "Статус устройства";
}

<h4>Статус устройства</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="StatusName" class="control-label"></label>
                <input asp-for="StatusName" class="form-control" />
                <span asp-validation-for="StatusName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Сохранить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.MobileStatus>

@{
    ViewData["Title"] = "Статус устройств";
}

<h1>Статус устройств</h1>

<p>
    <a asp-action="Create">Добавить статус устройства</a>
</p>
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.StatusName)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.StatusName)
                </td>
            </tr>
        }
    </tbody>
</table>

```

```

        </th>
        <th></th>
    </tr>
</thead>
<tbody>
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.StatusName)
        </td>
        <td>
            <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
            <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
            <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
        </td>
    </tr>
}
</tbody>
</table>
@model WebAppMobileRecord.Data.MobileType

@{
    ViewData["Title"] = "Тип устройства";
}
<h4>Тип устройства</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="TypeName" class="control-label"></label>
                <input asp-for="TypeName" class="form-control" />
                <span asp-validation-for="TypeName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model WebAppMobileRecord.Data.MobileType

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

<h3>Вы уверены что хотите удалить?</h3>
<div>
    <h4>Тип устройства</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.TypeName)
        </dt>
    </dl>

```



```

        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.TypeName)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Удалить" class="btn btn-primary" /> |
        <a asp-action="Index">Вернуться к списку</a>
    </form>
</div>

@model WebAppMobileRecord.Data.MobileType

@{
    ViewData["Title"] = "Тип устройства";
}

<div>
    <h4>Тип устройства</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.TypeName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.TypeName)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>
@model WebAppMobileRecord.Data.MobileType

@{
    ViewData["Title"] = "Тип устройства";
}

<h4>Тип устройства</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="TypeName" class="control-label"></label>
                <input asp-for="TypeName" class="form-control" />
                <span asp-validation-for="TypeName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Сохранить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

```

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.MobileType>

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

<h1>Типы устройств</h1>

<p>
    <a asp-action="Create">Добавить новый тип устройства</a>
</p>
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.TypeName)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.TypeName)
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
                    <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
                    <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
                </td>
            </tr>
        }
    </tbody>
</table>

@model WebAppMobileRecord.Data.OSType

@{
    ViewData["Title"] = "Тип ОС";
}

<h4>Тип ОС</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="OSTypeName" class="control-label"></label>
                <input asp-for="OSTypeName" class="form-control" />
                <span asp-validation-for="OSTypeName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

```

```

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model WebAppMobileRecord.Data.OSType

@{
    ViewData["Title"] = "Тип ОС";
}

<h3>Вы уверены что хотите удалить?</h3>
<div>
    <h4>Тип ОС</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.OSTypeName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.OSTypeName)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Удалить" class="btn btn-primary" /> |
        <a asp-action="Index">Вернуться к списку</a>
    </form>
</div>
@model WebAppMobileRecord.Data.OSType

@{
    ViewData["Title"] = "Тип ОС";
}

<div>
    <h4>Тип ОС</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.OSTypeName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.OSTypeName)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>

@model WebAppMobileRecord.Data.OSType

@{
    ViewData["Title"] = "Тип ОС";
}

<h4>Тип ОС</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="OSTypeName" class="control-label"></label>
        <input asp-for="OSTypeName" class="form-control" />
        <span asp-validation-for="OSTypeName" class="text-danger"></span>
      </div>
      <div class="form-group">
        <input type="submit" value="Сохранить" class="btn btn-primary" />
      </div>
    </form>
  </div>
</div>

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

@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.OSType>

@{
  ViewData["Title"] = "Тип ОС";
}

<h1>Типы ОС</h1>

<p>
  <a asp-action="Create">Добавить тип ОС</a>
</p>
<table class="table table-striped">
  <thead>
    <tr>
      <th>
        @Html.DisplayNameFor(model => model.OSTypeName)
      </th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model) {
      <tr>
        <td>
          @Html.DisplayFor(modelItem => item.OSTypeName)
        </td>
        <td>
          <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
          <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
          <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
        </td>
      </tr>
    }
  </tbody>
</table>
@model WebAppMobileRecord.Data.OSVersion

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

```

```

}

<h4>Версия ОС</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="Version" class="control-label"></label>
                <input asp-for="Version" class="form-control" />
                <span asp-validation-for="Version" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OSTypeId" class="control-label"></label>
                <select asp-for="OSTypeId" class="form-control" asp-
items="ViewBag.OSTypeId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Создать" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model WebAppMobileRecord.Data.OSVersion

@{
    ViewData["Title"] = "Delete";
}
<h3>Вы уверены что хотите удалить?</h3>
<div>
    <h4>Версия ОС</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Version)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Version)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.OSType)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.OSType.Id)
        </dd class>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Удалить" class="btn btn-primary" /> |
        <a asp-action="Index">Вернуться к списку</a>
    </form>
</div>
@model WebAppMobileRecord.Data.OSVersion

```

```

@{
    ViewData["Title"] = "Версия ОС";
}

<div>
    <h4>Версия ОС</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Version)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Version)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.OSType)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.OSType.Id)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>
@model WebAppMobileRecord.Data.OSVersion

@{
    ViewData["Title"] = "Версия ОС";
}

<h4>Версия ОС</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="Version" class="control-label"></label>
                <input asp-for="Version" class="form-control" />
                <span asp-validation-for="Version" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OSTypeId" class="control-label"></label>
                <select asp-for="OSTypeId" class="form-control" asp-
items="ViewBag.OSTypeId"></select>
                <span asp-validation-for="OSTypeId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Сохранить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

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

@section Scripts {

```

```

    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.OSVersion>

@{
    ViewData["Title"] = "Версия ОС";
}

<h1>Версии ОС</h1>

<p>
    <a asp-action="Create">Добавить новую версию</a>
</p>
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Version)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.OSType)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Version)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.OSType.OSTypeName)
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
                    <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
                    <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
                </td>
            </tr>
        }
    </tbody>
</table>
@model string

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

<form asp-action="Create" method="post">
    <div class="form-group">
        <label for="name">Новая роль</label>
        <input name="name" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary">Добавить</button>
</form>
@model IEnumerable<Microsoft.AspNetCore.Identity.IdentityRole>

<h1>Список ролей</h1>

<p>
    <a asp-action="Create">Добавить роль</a>
</p>

```

```

<table class="table table-striped">
  <thead>
    <tr>
      <th>
        Наименование
      </th>
      <th>
      </th>
    </tr>
  </thead>
  <tbody>

    @foreach (var role in Model)
    {
      <tr>
        <td>@role.Name</td>
        <td>
          <a asp-action="Delete" asp-route-name="@role.Name">Удалить</a>
        </td>
      </tr>
    }
  </tbody>
</table><!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - Учет мобильных устройств</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" /><link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.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-fluid">
        <a class="navbar-brand" asp-area="" asp-controller="Mobiles" asp-
action="Index"><i class="bi bi-phone-fill"></i>Учет мобильных устройств</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" />
          <partial name="_Navigation" />
        </div>
      </div>
    </nav>
  </header>
  <div class="container-fluid">
    <main role="main" class="pb-3">
      @RenderBody()
    </main>
  </div>

  <footer class="border-top footer text-muted">
    <div class="container-fluid">
      &copy; 2021 - Учет мобильных устройств - Третьякова Анастасия
    </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>
@using Microsoft.AspNetCore.Identity
@using WebAppMobileRecord.Data
@inject SignInManager<AppUser> SignInManager
@inject UserManager<AppUser> UserManager

<ul class="navbar-nav">
    @if (SignInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a class="nav-link text-dark" href="#">Здравствуй,
@UserManager.FindByNameAsync(UserManager.GetUserName(User)).Result.FullName!</a>
        </li>
        <li class="nav-item">
            <form class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-
route-returnUrl="@Url.Action("Index", "Mobiles", new { area = "" })">
                <button type="submit" class="nav-link btn btn-link text-
dark">Выйти</button>
            </form>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="Identity" asp-
page="/Account/Login">Вход</a>
        </li>
    }
</ul>
@using Microsoft.AspNetCore.Identity
@using WebAppMobileRecord.Data
@inject SignInManager<AppUser> SignInManager
@inject UserManager<AppUser> UserManager

<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="Mobiles" asp-
action="Index">Устройства</a>
        </li>
        @if (User.IsInRole("Администратор"))
        {
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-expanded="false">
                    Данные
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" asp-area="" asp-controller="OSTypes" asp-
action="Index">Тип ОС</a>
                    <a class="dropdown-item" asp-area="" asp-controller="OSVersions" asp-
action="Index">Версии устройств</a>
                    <a class="dropdown-item" asp-area="" asp-controller="MobileTypes"
asp-action="Index">Тип устройств</a>
                    <a class="dropdown-item" asp-area="" asp-controller="MobileStatus"
asp-action="Index">Статусы устройств</a>
                </div>
            </li>
        }
    }

```

```

        <a class="dropdown-item" asp-area="" asp-controller="Vendors" asp-
action="Index">Компании</a>
        @* <a class="dropdown-item" asp-area="" asp-
controller="AssignMobileIdentities" asp-action="Index">Назначения</a> *@
    </div>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="AppUsers" asp-
action="Index">Сотрудники</a>
    </li>

    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Roles" asp-
action="Index">Роли</a>
    </li>
}
}
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Description">О Программе</a>
</li>
</ul>
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>

@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}

<h1 class="text-danger">Ошибка.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed
    information about the error that occurred.
</p>
<p>
    <strong>The Development environment shouldn't be enabled for deployed
    applications.</strong>
    It can result in displaying sensitive information from exceptions to end users.
    For local debugging, enable the <strong>Development</strong> environment by setting
    the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
    <strong>Development</strong>
    and restarting the app.
</p>
@model WebAppMobileRecord.Data.Vendor
@{
    ViewData["Title"] = "Мобильный поставщик";
}

<h4>Мобильный поставщик</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="VendorName" class="control-label"></label>
        <input asp-for="VendorName" class="form-control" />
        <span asp-validation-for="VendorName" class="text-danger"></span>
      </div>
      <div class="form-group">
        <input type="submit" value="Создать" class="btn btn-primary" />
      </div>
    </form>
  </div>
</div>

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

@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model WebAppMobileRecord.Data.Vendor

@{
  ViewData["Title"] = "Мобильный поставщик";
}

<h3>Вы уверены что хотите удалить?</h3>
<div>
  <h4>Мобильный поставщик</h4>
  <hr />
  <dl class="row">
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.VendorName)
    </dt>
    <dd class = "col-sm-10">
      @Html.DisplayFor(model => model.VendorName)
    </dd>
  </dl>

  <form asp-action="Delete">
    <input type="hidden" asp-for="Id" />
    <input type="submit" value="Удалить" class="btn btn-danger" /> |
    <a asp-action="Index">Вернуться к списку</a>
  </form>
</div>
@model WebAppMobileRecord.Data.Vendor

@{
  ViewData["Title"] = "Мобильный поставщик";
}

<div>
  <h4>Мобильный поставщик</h4>
  <hr />
  <dl class="row">
    <dt class = "col-sm-2">
      @Html.DisplayNameFor(model => model.VendorName)
    </dt>
    <dd class = "col-sm-10">

```

```

        @Html.DisplayFor(model => model.VendorName)
    </dd>
</dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Редактировать</a> |
    <a asp-action="Index">Вернуться к списку</a>
</div>
@model WebAppMobileRecord.Data.Vendor

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

<h4>Мобильный поставщик</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="VendorName" class="control-label"></label>
                <input asp-for="VendorName" class="form-control" />
                <span asp-validation-for="VendorName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Сохранить" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<WebAppMobileRecord.Data.Vendor>

@{
    ViewData["Title"] = "Index";
}
<h1>Мобильный поставщик</h1>
<p>
    <a asp-action="Create">Добавить поставщика</a>
</p>
<table class="table table-striped">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.VendorName)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.VendorName)
                </td>
            </tr>
        }
    </tbody>
</table>

```

```

        <td>
            <a asp-action="Edit" asp-route-id="@item.Id">Редактировать</a> |
            <a asp-action="Details" asp-route-id="@item.Id">Детальная форма</a> |
            <a asp-action="Delete" asp-route-id="@item.Id">Удалить</a>
        </td>
    </tr>
}
</tbody>
</table>
@using WebAppMobileRecord
@using WebAppMobileRecord.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@{
    Layout = "_Layout";
}
@model WebAppMobileRecord.Data.Mobile

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

<h4>Устройства</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="View">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Number" class="control-label"></label>
                <input asp-for="Number" class="form-control" />
                <span asp-validation-for="Number" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="AddedDate" class="control-label"></label>
                <input asp-for="AddedDate" class="form-control" />
                <span asp-validation-for="AddedDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DeactivatedDate" class="control-label"></label>
                <input asp-for="DeactivatedDate" class="form-control" />
                <span asp-validation-for="DeactivatedDate" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="OSVersionId" class="control-label"></label>
                <select asp-for="OSVersionId" class="form-control" asp-
items="ViewBag.OSVersionId"></select>
            </div>
            <div class="form-group">
                <label asp-for="MobileStatusId" class="control-label"></label>
                <select asp-for="MobileStatusId" class="form-control" asp-
items="ViewBag.MobileStatusId"></select>
            </div>
            <div class="form-group">
                <label asp-for="MobileTypeId" class="control-label"></label>
                <select asp-for="MobileTypeId" class="form-control" asp-
items="ViewBag.MobileTypeId"></select>
            </div>
            <div class="form-group">
                <label asp-for="VendorId" class="control-label"></label>
                <select asp-for="VendorId" class="form-control" asp-
items="ViewBag.VendorId"></select>
            </div>
            <div class="form-group">

```

```

        <input type="submit" value="Создать" class="btn btn-primary" />
    </div>
</form>
</div>
</div>
<div>
    <a asp-action="Index">Вернуться к списку</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
{
    "ConnectionStrings": {
        "DefaultConnection": "Server=LAPTOP-
64SU0IQ4\\SQLEXPRESS;Database=WebAppMobileRecord;Trusted_Connection=True;MultipleActiveRe
sultSets=true"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft": "Warning",
            "Microsoft.Hosting.Lifetime": "Information"
        }
    },
    "AllowedHosts": "*"
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
namespace WebAppMobileRecord
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using WebAppMobileRecord.Data;

```

```

using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
namespace WebAppMobileRecord
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
        container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(
                    Configuration.GetConnectionString("DefaultConnection")));
            services.AddDefaultIdentity<AppUser>(options =>
options.SignIn.RequireConfirmedAccount = false)
                .AddRoles<IdentityRole>()
                .AddEntityFrameworkStores<ApplicationDbContext>();
            // services.AddIdentity<AppUser, IdentityRole>(options =>
options.SignIn.RequireConfirmedAccount = true)
            //     .AddEntityFrameworkStores<ApplicationDbContext>();
            services.AddControllersWithViews();
            services.AddRazorPages();
        }
        // This method gets called by the runtime. Use this method to configure the HTTP
        request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseDatabaseErrorPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to change this for
                production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Mobiles}/{action=Index}/{id?}");
                endpoints.MapRazorPages();
            });
        }
    }
}

```

<i>Обозначение</i>	<i>Наименование</i>	<i>Примечание</i>
	<i>Текстовые документы</i>	
<i>БГУИР ДП 1-40 01 01 095 ПЗ</i>	<i>Пояснительная записка</i>	<i>145 с.</i>
	<i>Отзыв руководителя</i>	
	<i>Рецензия</i>	
	<i>Отчёт о проверке на заимствования</i>	
	<i>Справка о внедрении</i>	
	<i>Графические документы</i>	
<i>ГУИР.881074.001 СА</i>	<i>Алгоритм проверки доступа на</i>	<i>Формат А1</i>
	<i>клиентской стороне</i>	
	<i>Схема алгоритма</i>	
<i>ГУИР.881074.002 СА</i>	<i>Алгоритм</i>	<i>Формат А1</i>
	<i>добавления</i>	
	<i>сотрудника</i>	
	<i>Схема алгоритма</i>	
<i>ГУИР.881074.003 СА</i>	<i>Алгоритм</i>	<i>Формат А1</i>
	<i>добавления</i>	
	<i>устройства</i>	
	<i>Схема алгоритма</i>	
<i>ГУИР.881074.001 ПЛ</i>	<i>Диаграмма вариантов</i>	<i>Формат А1</i>
	<i>использования</i>	
	<i>Плакат</i>	
<i>ГУИР.881074.002 ПЛ</i>	<i>Схема базы данных</i>	<i>Формат А1</i>
	<i>Плакат</i>	
<i>ГУИР.881074.003 ПЛ</i>	<i>Диаграмма классов</i>	<i>Формат А1</i>
	<i>Плакат</i>	

					БГУИР ДП 1-40 01 01 095 Д1			
<i>Изм</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>	<i>Веб-приложение для учета мобильных устройств ЗАО «Кьюликс Системс» на языке программирования С# Ведомость дипломного проекта</i>	<i>Лит.</i>	<i>Лист</i>	<i>Листов</i>
<i>Разраб.</i>		<i>А.В. Третьякова</i>						
<i>Пров.</i>		<i>Д.В. Горбачев</i>				Т	145	145
<i>Т.контр.</i>		<i>Д.В. Горбачев</i>				<i>ПОИТ, гр. 881074</i>		
<i>Н.контр.</i>		<i>С.В. Болтак</i>						
<i>Утв.</i>								