



## Содержание

Введение	3
1 Объектно-ориентированный анализ и проектирование приложения	4
1.1 Назначение и цели создания Web-приложения	4
1.2 Проектирование модели	5
2 Проектирование Web-приложения	9
2.1 Требования к Web-приложению	9
2.2 Структура Web-приложения	9
2.3 Проектирование макета Web-приложения	12
2.4 Программно-технические средства, необходимые для разработки приложения	14
2.5 Защита и сохранность данных	16
2.6 Организация и ведение информационной базы(модели)	16
3 Реализация Web-приложения	22
3.1 Разработка административной части приложения	22
3.2 Разработка клиентской части приложения	23
3.3 Описание используемых функций и процедур	24
4 Описание Web-приложения	30
4.1 Общие сведения	30
4.2 Функциональное назначение	30
4.3 Описание разделов сайта	30
5 Методика испытаний	31
5.1 Технические требования	31
5.2 Функциональное тестирование	31
6 Применение	37
6.1 Назначение приложения	37
6.2 Программно-аппаратное обеспечение сервера и клиента	37
Заключение	38
Список информационных источников	39
Приложение А. Текст программных модулей	40

					ОП Т.795007					
Изм.	Лист	№ докум.	Подпись	Дата	Отчет по преддипломной практике			Лит.	Лист	Листов
Разраб.		Выговская Е.Р.								
Провер.		Михалевич В.Ю.							2	74
Т. контр.								КБП		
Н. контр.										
Утверд.										

## Введение

Web-приложения – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Сегодня Web-приложения набирают популярность. Самыми посещаемыми сайтами становятся не чисто информационные, гипертекстовые сайты, а те, которые предоставляют какой-либо сервис, как-либо взаимодействуют с пользователем. Но даже и обычные информационные сайты часто используют системы управления контентом для удобства управления информацией, так что и их тоже можно причислить к веб-приложениям.

Задачей на преддипломную практику является реализация web-приложения для подбора персонала на проекты, которое будет предоставлять возможность поиска необходимых людей для создания проектов, возможность удобного просмотра всех людей, возможность загрузки резюме сотрудника в базу, а также удобное отображение. Также будет обеспечено хранение необходимой информации в базе данных.

Отчет по преддипломной практике состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации программного приложения.

В первом разделе «Объектно-ориентированный анализ и проектирование приложения» раскрывается организационная сущность задачи, описывается предметная область, и назначение и цели создания Web-приложения. Описываются новые возможности Web-приложения, а также ее отличия от предыдущих версий, перечисляются основные функции приложения. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи.

Во втором разделе «Проектирование Web-приложения» перечисляются требования к Web-приложению, его структура проектирование макета, программно-технические средства необходимые для разработки, защита и сохранность данных, организация и ведение информационной базы (модели).

В третьем разделе «Реализация Web-приложения» описываются этапы разработки административной и клиентской части Web-приложения. Также описываются используемые функции и процедуры.

В четвертом разделе «Описание Web-приложения» приводятся общие сведения о приложении, ее функциональное назначение и описание разделов сайта.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к совместимости сайта с различными браузерами разными настройками браузеров. Представляются результаты функционального тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении сайта и области его применения классе решаемых задач, ограничениях, накладываемых на область применения, требования к техническим и программным ресурсам, организация диалога с пользователем, способы ввода информации.

В заключении будет проанализирована поставленная задача, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст модулей Web-приложения.

# 1 Объектно-ориентированный анализ и проектирование приложения

## 1.1 Назначение и цели создания Web-приложения

Современные компьютерные программы, онлайн-сервисы и приложения создают максимально простую и свободную среду для общения клиента и исполнителя (продавца). Ставший популярным термин «управление взаимоотношения с клиентами», и есть та самая вспомогательная система, позволяющая персонализировать каждого клиента (проект) и увеличить продажи за счет работы с лидами.

Web-приложение для подбора персонала на проекты направлено на упрощение работы менеджера проектов, который ищет по резюме необходимых сотрудников для создания программных проектов для заказчиков.

Менеджер проектов – это специалист, чьей главной задачей является управление проектом в целом: проектирование и расстановка приоритетов, планирование выполнения задач, контроль, коммуникации, а также оперативное решение проблем.

Резюме – это собирательный документ, в котором указывается все этапы обучения и работы, и можно также указать достижения и личные качества, о которых не можете рассказать с помощью других документов. Оно всегда предполагает указание всей информации в довольно строгом формате и последовательности.

Программный проект – это комплект официальных документов, позволяющий другим специалистам воспроизводить, или тиражировать, или развивать программное изделие без присутствия автора.

Заказчик – лицо (физическое или юридическое), заинтересованное в выполнении исполнителем работ, оказании им услуг или приобретении у продавца какого-либо продукта (в широком смысле). Иногда при этом предполагается оформление заказа, но не обязательно.

Сотрудник – лицо, осуществляющее должностные обязанности совместно с кем-либо при выполнении поставленных задач. Данный труд может оплачиваться (научный сотрудник) либо быть безвозмездным (в редких случаях).

Чтобы пользоваться приложением нужно пройти регистрацию – процесс, где менеджер проектов вводит почту и пароль, после которого ему предоставляются права на пользование приложением. После регистрации менеджер проектов становится пользователем и может зайти на сайт после того, как пройдет авторизацию.

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

У проектируемого Web-приложения на первый план выходит взаимодействие человек-компьютер. Перед тем как рисовать экраны и именовать разделы, надо определиться с тем, как человек будет пользоваться приложением. Всегда, проектируя интерфейс сайта, важно сделать его не перегруженным и с простым, для применения, функционала.

На основе предметной области можно выделить такие функции как:

- авторизация менеджера;
- добавление нового сотрудника в базу;
- редактирование информации о новом сотруднике;
- загрузка и отображение резюме сотрудников;
- просмотр доступных сотрудников для проектов.

В настоящее время с поставленными задачами предложенные решения на рынке не выполняют весь функционал приложения подобного типа. У аналога данного программного средства Соорег отсутствует просмотр резюме сотрудников.

## 1.2 Проектирование модели

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD), которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы. Основными понятиями данной нотации являются понятия сущности и связи. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Связь определяется как отношение или некоторая ассоциация между отдельными сущностями.

Графическая модель данных строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей. Информационная модель базы представлена на диаграмме «Сущность-связь» на рисунке 1.

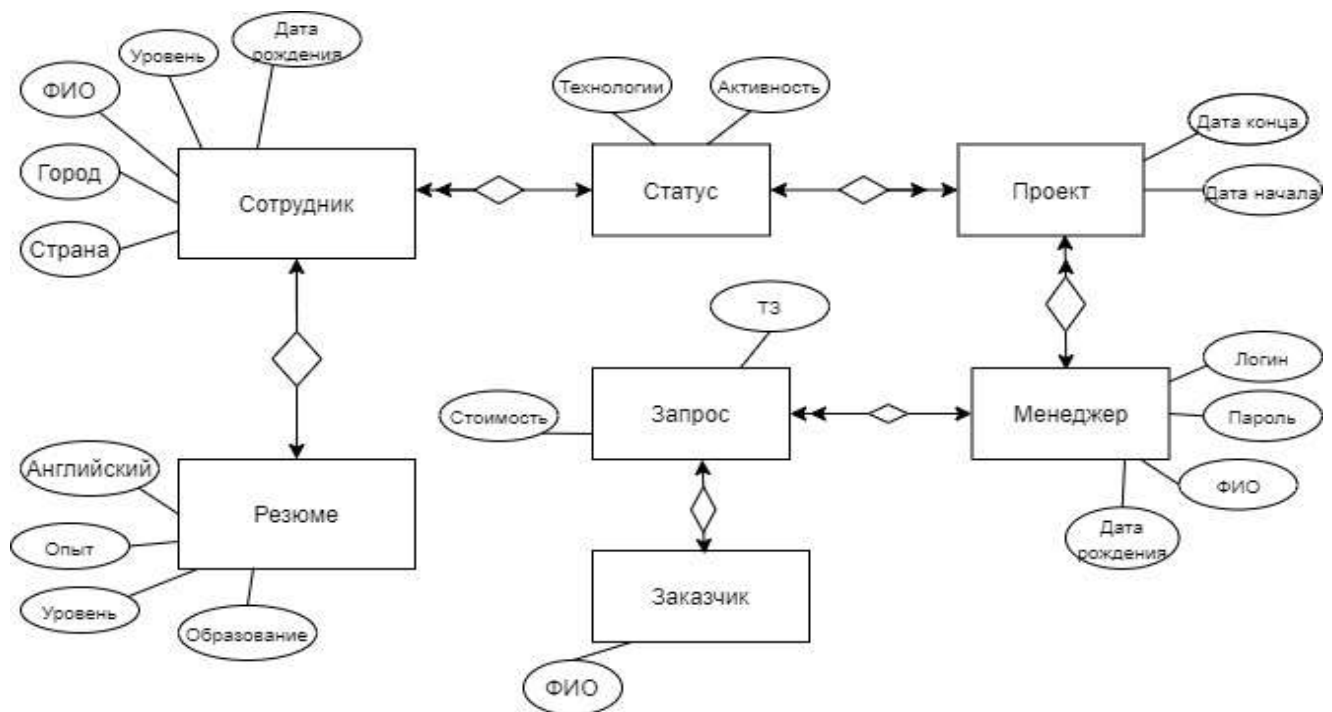


Рисунок 1

Исходя из предметной области можно выделить следующие сущности разработки: «Сотрудник», «Статус», «Проект», «Менеджер», «Запрос», «Заказчик», «Резюме».

Для сущности «Сотрудник» атрибутами будут являться:

- страна;
- город;
- фео;

- уровень;
- дата рождения.

Для сущности «Статус» атрибутами будут являться:

- технологии;
- активность.

Для сущности «Проект» атрибутами будут являться:

- дата начала;
- дата конца.

Для сущности «Менеджер» атрибутами будут являться:

- фιο;
- логин;
- пароль;
- дата рождения.

Для сущности «Запрос» атрибутами будут являться:

- тз;
- стоимость.

Для сущности «Заказчик» атрибутом будет ФИО.

Для сущности «Резюме» атрибутами будут являться:

- английский;
- опыт;
- уровень;
- образование

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества так называемых вариантов использования, предоставляемых системой множеству актеров или сущностей, взаимодействующих с системой. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером.

Варианты использования определяют функциональные возможности. Каждый из них представляет определенный способ использования. Таким образом, каждый вариант использования соответствует последовательности действий для того, чтобы клиент мог получить определенный результат.

Исходя из диаграммы вариантов использования можно выделить следующие «includes» функции, которые включают в авторизацию такие возможности как:

- добавление сотрудника;
- авторизация менеджера;
- просмотр сотрудников;
- просмотр заказчиков;
- поиск заказчиков;
- изменение данных в базе данных.

Также эти функции расширяются с помощью «extend» функции – обработки запроса на сервере.

Диаграмма вариантов использования представлена в графической части на листе 1.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности.

Для разработки Web-приложения используется паттерн MVC. MVC – это паттерн проектирования веб-приложений, который включает в себя несколько более мелких шаблонов. При использовании MVC на три отдельных компонента разделены модель данных приложения, пользовательский интерфейс и логика взаимодействия пользователя с системой, благодаря чему модификация одного из этих компонентов оказывает минимальное воздействие на остальные или не оказывает его вовсе.

Windows 10 – является наиболее популярной и удобной операционной системой. Система призвана стать единой для разных устройств, таких как персональные компьютеры, планшеты, смартфоны, консоли и т.д.

Данное web-приложение разрабатывается в среде программирования Visual Studio 2019 и Visual Studio Code 1.45. Они содержат богатый набор различных типов данных и компонентов, облегчающих создание программного продукта под Windows.

Visual Studio – это набор инструментов и средств, предназначенных для разработчиков программ, с широким набором поддерживаемых языков программирования. 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 (как, например, 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 обеспечения [11].

Google Chrome – браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink (до апреля 2013 года использовался WebKit). Первая публичная бета-версия для Windows вышла 2 сентября 2008 года, а первая стабильная – 11 декабря 2008 года. По данным StatCounter, Chrome используют около 300 миллионов интернет-пользователей, что делает его самым популярным браузером в мире – его рыночная доля на сентябрь 2018 года составляет 60.6 % [13].

Язык программирования C# – поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C# сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником – языком C, – наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основным используемым языком запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Draw.io – это приложение на диске Google для создания диаграмм, которое позволяет рисовать:

- блок-схемы;

- UML;
- диаграммы сущность-связь;
- сетевые диаграммы;
- модели бизнес-процессов;
- организационные схемы;
- электрические схемы;
- каркасные схемы и модели.

Язык запросов SQL – это запросы, которые составляются (программистами) из последовательности SQL – инструкций. Эти инструкции задают, что надо сделать с входным набором данных для генерации выходного набора. Все запросы Access строит на основе SQL.

Структура Web-приложения представлена в графической части на листе 2.

Блок-схема алгоритма – графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и блоков – графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия. В данном приложении блок схема представляет собой алгоритм пошаговой авторизации в приложении. На первом шаге пользователь сталкивается с авторизацией, где предстоит заполнить логин и пароль. Далее происходит вход в приложении и отображение главной страницы. Вся блок-схема алгоритма представлена на в графической части на листе 3.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Приложение будет разделено на две родительские папки «DataModel» и «ExtResBlzr», в которых находятся папки и файлы отвечающие за клиентскую и серверную разработку соответственно. Основная часть компонентов представлена с расширением «.cs», так же имеются документы с расширениями «.razor», «.json», «.html». Вид диаграммы компонентов для данной проектируемой системы представлен в графической части на листе 4.



## 2 Проектирование Web-приложения

### 2.1 Требования к Web-приложению

Стилистическое оформление Web-приложения должно соответствовать техническому заданию. После согласования графического эскиза сайта с преподавателем, переделка стилистического оформления Web-приложения невозможна (за исключением незначительных поправок).

Требования к графическому дизайну:

- дизайн Web-приложения должен соответствовать техническому заданию;
- дизайн Web-приложения должен быть лаконичным и в то же время выглядеть стильно, современно;
- приветствуется использование небольших, но стильных графических элементов (пиктограмм) в оформлении контента Web-приложения.

Требования к шрифтовому оформлению:

- размер (кегель) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана;
- предпочтителен шрифт без засечек.

Web-приложение должно обеспечивать корректное отображение данных в следующих браузерах:

- Internet Explorer;
- Opera;
- Mozilla Firefox;
- Google Chrome.

Система управления контентом Web-приложения должна обеспечить пользователю Web-приложения возможность выполнения следующих действий:

- реализовать регистрацию и авторизацию в приложение;
- осуществить возможность создания нового сотрудника;
- реализовать просмотр всех сотрудников;
- организовать удобное отображение информации о партнере и сотруднике;
- возможность просмотра резюме в формате .pdf;
- обеспечить хранение необходимой информации в базе данных.

Компоновка страниц Web-приложения должна обеспечивать автоматическое масштабирование страниц в зависимости от ширины рабочего поля браузера пользователя. Минимальное разрешение экрана, при котором необходимо обеспечить полноценное отображение страниц (без полосы горизонтальной прокрутки), составляет 1024x768. Также сайт должен адекватно выглядеть при разрешении 1024x768.

### 2.2 Структура Web-приложения

Логическая структура web-приложения отображает то, как именно связаны между собой страницы одного сайта. При разработке приложения для управления коллекциями максимальная связь страниц достигнута при помощи постоянного отображения навигационной панели сайта, благодаря этому навигация по сайту не вызывает никаких затруднений. Главная страница сайта имеет определённую структуру, представленную в соответствии с рисунком 2.



Рисунок 2

Сайт вместе с его административным разделом, не заполненный контентом, состоит из двадцати .cs файлов, десяти .razor файлов, пяти .json файлов, двух главных каталога сервера (DataModel) и клиента (ExtResBlzr), которые в свою очередь разбиты на подкаталоги. Все файлы и папки, относящиеся к серверной части раздела находятся в папке «DataModel».

Серверная часть раздела Web-приложения состоит из папок:

- «bin», в этой папке хранятся все файлы исполняемого приложения;
- «obj», в этой папке хранятся файлы с функцией промежуточной обработки;
- «Models», в этой папке хранятся файлы представляющие собой модели данных.

В корневой директории «DataModel» находятся тринадцать файлов. Описание физической структуры серверной части приведено в таблице 1.

Таблица 1 – Описание физической структуры серверной части

Имя файла	Директория	Описание
Document.cs	\DataModel\Models	Модель данных документа
Patner.cs	\DataModel\Models	Модель данных партнера
Resource.cs	\DataModel\Models	Модель данных ресурса
Request.cs	\DataModel\Models	Модель данных запроса
Contact.cs	\DataModel\Models	Модель данных контакта
PartnerDocument.cs	\DataModel\Models	Модель данных документа партнера
ResourceDocument.cs	\DataModel\Models	Модель данных документа ресурса
FavoriteResource.cs	\DataModel\Models	Модель данных любимого ресурса
project.assets.json	\DataModel\obj	Функции промежуточной обработки
DataModel.deps.json	\DataModel\bin	Подключенные библиотеки
.gitignore	\DataModel	Настройки сокрытия системы контроля версий git

Продолжение таблицы 1

Имя файла	Директория	Описание
IContactRepository.cs	\DataModel	Интерфейс контакта
ParnterRepository.cs	\DataModel	Функции для партнеров
package-lock.json	\DataModel	Скрипты всех подгружаемых пакетов
package.json	\DataModel	Скрипты основных настроек

Файлы и папки, относящиеся к клиентской части находятся в корневом каталоге Web-приложения, который состоит из папок:

- «wwwroot», в этой папке хранятся все файлы исполняемого приложения;
- «Controllers», в этой папке хранятся HTTP-запросы;
- «Pages», в этой папке находятся все страницы приложения;
- «Data», в этой папке хранятся данные из базы данных.

В корневой директории «ExtResBlzr» находятся двенадцать файлов. Описание физической структуры клиентской части приведено в таблице 2.

Таблица 2 – Описание физической структуры клиентской части

Имя файла	Директория	Описание
ContactController.cs	\ExtResBlzr\ Controller	Контроллер контакта
ResourceController.cs	\ExtResBlzr\ Controller	Контроллер ресурса
AppDbContext.cs	\ExtResBlzr\Data	Все данные из базы данных
ContactList.razor	\ExtResBlzr\Pages	Список всех контактов
CreateContact.razor	\ExtResBlzr\Pages	Создание контакта
Error.cshtml.cs	\ExtResBlzr\Pages	Форма появления ошибки
package-lock.json	\ExtResBlzr\Pages	Скрипты всех подгружаемых пакетов клиента
appSettings.json	\ExtResBlzr\Pages	Скрипт подгружаемых библиотек
Startup.cs	\ExtResBlzr	Точка входа в приложение
Program.cs	\ExtResBlzr	Создается по умолчанию при входе в приложение
ExtResBlzr.sln	\ExtResBlzr	Файл организации проекта в Visual Studio
bootstrap.min.cs	\ExtResBlzr\ wwwroot	Скрипт подгружаемых стилей

## 2.3 Проектирование макета Web-приложения

Дизайн сайта будет «адаптивным» – вариант веб-дизайна, при котором ширина страницы сайта будет адаптирована под любое разрешение. То есть на мониторах с большим разрешением страница будет выводиться на весь экран.

Так как большая часть информации будет представлена не текстом, а навигационными элементами, то для удобства восприятия был выбран именно этот тип. Также в дизайне сайта широкое распространение получили компоненты, подгруженные из css библиотеки «Materialize» из-за того, что информация, помещаемая на сайт, будет структурирована, а компоненты позволяют легко представить структурированную информацию в удобном для восприятия виде.

На главной странице будет расположена навигационная панель со ссылками на подстраницы, а также главная область – область поиска подходящего сотрудника. Макет главной страницы представлен ниже на рисунке 3.

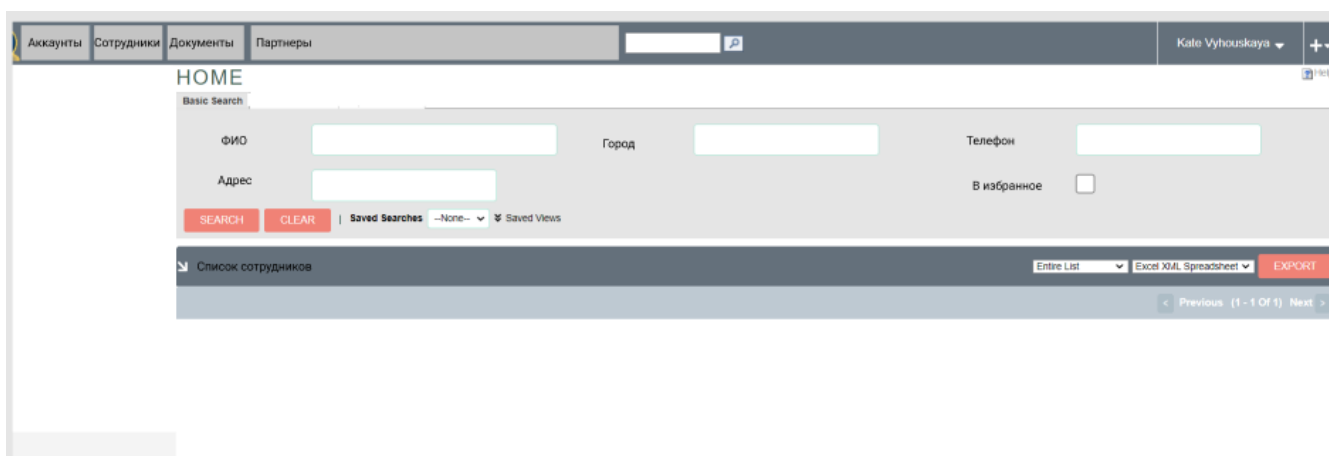


Рисунок 3

На странице «Сотрудники» будет доступен список всех сотрудников, представленный на рисунке 4, а также возможность создания сотрудника, макет которого представлен на рисунке 4.

## CONTACTS

SAVE

CANCEL

### CONTACT INFORMATION

Имя

Фамилия

Логин

SELECT

CLEAR

Уровень

--None--

Доп. инф

Технологии

Дата рождения

(MM/DD/YYYY)

SELECT

CLEAR

Активность

☐

Не беспокоить

☒

Менеджер

Kate Vyhouskaya

SELECT

CLEAR

Рисунок 4

Страница «Документы» будет представлять собой страницу, на которой находится вся необходимая информация о резюме всех сотрудников, а также возможность добавления нового резюме, макет представлен на рисунке 5.

Документы

SAVE CANCEL

Название

Резюме

Тип документа

Category Value

Статус

Менеджер

Дата загрузки

Заголовок

Описание

Версия

Активность

Категория

Степень важности

Дата

SELECT CLEAR

SELECT CLEAR

Рисунок 5

## 2.4 Программно-технические средства, необходимые для разработки приложения

Инструментами разработки будут являться:

- операционная система Windows 10 Professional;
- среда разработки Visual Studio Code;
- среда разработки Visual Studio 2019;
- фреймворк Blazor;
- язык программирования C#;
- система управления реляционными базами данных Microsoft SQL Server 2017;
- браузер GoogleChrome;
- язык разметки HTML;
- язык стилей CSS;
- программа для построения диаграмм Draw.io.

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами операционная система – это набор программного обеспечения, который обеспечивает работу компьютера. Основными функциями операционной системы являются:

- управление файловой системой (запись, изменение, копирование файлов, контроль доступа);
- управление выполнением программ (распределение процессорного времени, загрузка программ с диска в оперативную память, перехват потенциально опасных действий);
- управление памятью (кэширование, распределение, контроль сохранности данных);
- диалог с пользователем (чтение команд с клавиатуры, с мыши, вывод информации на экран, на принтер).

Windows 10 – является наиболее популярной и удобной операционной системой. Система призвана стать единой для разных устройств, таких как персональные компьютеры, планшеты, смартфоны, консоли и т.д [5].

HTML (HyperText Markup Language) - язык разметки гипертекста – предназначен для создания Web-страниц. CSS (Cascading Style Sheets – Каскадные Таблицы Стилей) – условный язык описания внешнего вида документа. В основном, конечно же, CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS) [1].

Blazor – это бесплатная веб-платформа с открытым исходным кодом, которая позволяет разработчикам создавать веб-приложения с использованием C# и HTML. Он разрабатывается Microsoft [10]. Приложение Blazor может взаимодействовать с JavaScript (причем оба они работают на стороне клиента), например, вызывать (повторно использовать) функции JavaScript из .NET методов.

Данное web-приложение разрабатывается в среде программирования Visual Studio 2019 и Visual Studio Code 1.45. Они содержат богатый набор различных типов данных и компонентов, облегчающих создание программного продукта под Windows.

Visual Studio – это набор инструментов и средств, предназначенных для разработчиков программ, с широким набором поддерживаемых языков программирования. 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 (как, например, 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 обеспечения [6].

Visual Studio Code – это редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS [7]. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Google Chrome – браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink (до апреля 2013 года использовался WebKit). Первая публичная бета-версия для Windows вышла 2 сентября 2008 года, а первая стабильная – 11 декабря 2008 года. По данным StatCounter, Chrome используют около 300 миллионов интернет-пользователей, что делает его самым популярным браузером в мире – его рыночная доля на сентябрь 2018 года составляет 60.6 % [11].

Язык программирования C# – поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя

распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности [9]. C# сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником – языком C, – наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основным используемым языком запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями [8]. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Draw.io – это приложение на диске Google для создания диаграмм [4], которое позволяет рисовать:

- блок-схемы;
- UML;
- диаграммы сущность-связь;
- сетевые диаграммы;
- модели бизнес-процессов;
- организационные схемы;
- электрические схемы;
- каркасные схемы и модели.

Язык запросов SQL – это запросы, которые составляются (программистами) из последовательности SQL – инструкций. Эти инструкции задают, что надо сделать с входным набором данных для генерации выходного набора. Все запросы Access строит на основе SQL.

## 2.5 Защита и сохранность данных

Конфиденциальность – обеспечение исключительно авторизованного доступа к информации: информация не должна предоставляться и не должна раскрываться неавторизованным физическим лицам, организациям или процессам. Целостность – поддержание и обеспечение точности и полноты данных на протяжении всего жизненного цикла: данные не должны быть изменены неавторизованным или незаметным способом. Доступность – обеспечение беспрепятственного доступа к информации авторизованным пользователям: системы хранения и обработки информации, интерфейсы работы с информацией, системы обеспечения авторизованного доступа и каналы связи должны функционировать корректно.

Для каждого пользователя существует отдельный аккаунт, что позволяет пользователям управлять только личными данными. Все пароли, хранящиеся в базе данных строго зашифрованы, что добавляет безопасности при случае взлома.

## 2.6 Организация и ведение информационной базы (модели)

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двумерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна





Продолжение таблицы 3

Имя поля	Тип поля	Размер поля, байт	Описание поля
Notes	String	100	Пометки
PartnerId	Integer	4	Партнер
DateofBirth	DateTime	–	Дата рождения
Gender	Integer	4	Пол сотрудника

Таблица «EngagementOfPartner» хранит информацию о начале проекта и структура приведена в таблице 4.

Таблица 4 – Структура таблицы «EngagementOfPartner»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор проекта
RateId	Integer	4	Идентификатор рейтинга
ProjectCode	String	100	Тех. задание к проекту
PartnerId	Integer	4	Идентификатор партнера
ProjectJira	String	100	Логин в Jira
PartnerManId	Integer	4	Менеджер, привлечший партнера
DateStart	DateTime	–	Дата начала проекта
DateFinish	DateTime	–	Дата окончания проекта
Notes	String	500	Дополнительная информация

Таблица «Partner» хранит информацию о партнере и структура приведена в таблице 5.

Таблица 5 – Структура таблицы «Partner»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор проекта
Name	String	100	Имя партнера
Rating	String	100	Рейтинг партнера
Description	String	500	Тех. задание

Продолжение таблицы 5

Имя поля	Тип поля	Размер поля, байт	Описание поля
Technologies	String	100	Технологии
Country	String	100	Страна проживания партнера
City	String	100	Город проживания партнера
BillingAddress	String	100	Адрес
NdaDate	DateTime	–	Дата принятия заказа
NdaYears	DateTime	–	Дата начала разработки
NonSolicitation	Integer	4	Идентификатор рекомендации

Таблица «NonSolicitationType» хранит информацию о рекомендациях к проекту и структура приведена в таблице 6.

Таблица 6 – Структура таблицы «NonSolicitationType»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор рекомендации
Name	String	500	Рекомендация

Таблица «PartnerDocument» хранит информацию о техническом задании проекта и структура приведена в таблице 7.

Таблица 7 – Структура таблицы «PartnerDocument»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор тех. задания
PartnerId	Integer	4	Идентификатор партнера
DocumentId	Integer	4	Идентификатор проекта
Notes	String	500	Описание тех. задания

Таблица «Resource» хранит информацию о резюме сотрудника и структура приведена в таблице 8.

Таблица 8 – Структура таблицы «Resource»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор резюме
FirstName	String	100	Имя сотрудника
LastName	String	100	Фамилия сотрудника
PartnerId	Integer	4	Идентификатор партнера
Title	String	200	Образование
About	String	500	Дополнительная информация
Technologies	String	100	Технологии
LevelDeclared	String	100	Заявленный уровень опыта
EnglishSpoken	Bool	4	Говорит на английском
EnglishFdback	String	100	Отзыв о знаниях английского
Available	Bool	4	Доступен ли сейчас
CVToolLink Master	String	100	Ссылка на резюме в формате .pdf
Added	DateTime	–	Дата добавления резюме
Updated	DateTime	–	Дата обновления резюме

Таблица «ResourceDocument» хранит информацию о документе резюме и структура приведена в таблице 9.

Таблица 9 – Структура таблицы «ResourceDocument»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор резюме
ResourceId	Integer	4	Идентификатор резюме
DocumentId	Integer	4	Идентификатор проекта
Notes	String	500	Тех. задание

Таблица «Request» хранит информацию о запросе на создание проекта и структура приведена в таблице 10.

Таблица 10 – Структура таблицы «Request»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор запроса
IssuerPartnerId	Integer	4	Идентификатор партнера
Notes	String	500	Тех. задание
IssuedDate	DateTime	–	Дата принятия запроса
StartDate	DateTime	–	Дата начала разработки
EndDate	DateTime	–	Дата окончания разработки
Approximate Duration	String	200	Продолжительность разработки проекта
Description	String	500	Тех. задание
Technologies	String	100	Технологии
EnglishSpoken	Bool	4	Говорит на английском
IsActive	Bool	4	Активен в данный момент
FinalNotes	String	500	Дополнительная информация
LinkJira	String	100	Ссылка на Jira
LinkCollab	String	100	Ссылка на просмотр проекта
ResourceLevel	String	100	Уровень опыта

Таблица «Document» хранит информацию о готовом проекте и структура приведена в таблице 11.

Таблица 11 – Структура таблицы «Document»

Имя поля	Тип поля	Размер поля, байт	Описание поля
Id	Integer	4	Идентификатор проекта
FileName	String	100	Название проекта
FileDate	DateTime	–	Дата окончания разработки
FileSize	Integer	4	Размер готового проекта
ContentType	String	100	Тип разработанного проекта
Notes	String	100	Дополнительная информация

## 3 Реализация Web-приложения

### 3.1 Разработка административной части приложения

Веб браузеры взаимодействуют с веб-серверами при помощи гипертекстового транспортного протокола (HTTP). Когда пользователь нажимает на ссылку на веб-странице, заполняет форму или запускает поиск, HTTP запрос отправляется из вашего браузера на целевой сервер.

HTTP – широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Запрос включает в себя URL, определяющий затронутый ресурс, метод, определяющий требуемое действие (например, получить, удалить или опубликовать ресурс) и может включать дополнительную информацию, закодированную в параметрах URL (пары поле-значение, оформленные как строка запроса), как POST запрос (данные, отправленные методом HTTP POST), или в куки-файлах.

Веб серверы ожидают сообщений с клиентскими запросами, обрабатывают их по прибытию и отвечают веб-браузеру при помощи ответного HTTP сообщения. Ответ содержит строку состояния, показывающую, был ли запрос успешным, или нет.

Тело успешного ответа на запрос может содержать запрашиваемые данные (например, новую HTML страницу, или изображение), который может отображаться через веб-браузер.

URL (или URL адрес) – это форма уникального адреса конкретного веб-ресурса в сети Интернет. Он может ссылаться на веб-сайт, какой-то индивидуальный документ или изображение. Пользователю Интернета нужно вставить этот код в поле поиска, чтобы найти нужный сайт, документ, папку или изображение. На простом языке это означает следующее: благодаря URL адресу пользователь узнает информацию о том, где находятся нужные ему данные.

На рисунке 7 представлен скриншот из редактора кода «Visual Studio», где представлен каталог «DataModel» в котором расположены такие каталоги как:

- «bin»;
- «Models»;
- «obj».



Рисунок 7

Папка bin содержит двоичные файлы, которые являются фактическим исполняемым кодом приложения.

Папка obj содержит объектные или промежуточные файлы, которые представляют собой скомпилированные двоичные файлы, которые еще не были связаны. По сути, это фрагменты, которые будут объединены для создания окончательного исполняемого файла. Компилятор генерирует один объектный файл для каждого исходного файла, и эти файлы помещаются в папку obj .

Каждая из этих папок далее подразделяется на папки Debug и Release , которые соответствуют конфигурациям сборки проекта. Два типа файлов, описанных выше, помещаются в соответствующую папку в зависимости от того, какой тип сборки выполняется. Это позволяет легко определить, какие исполняемые файлы построены с отладочными символами, а какие были построены с включенной оптимизацией и готовы к выпуску.

В каталоге «Models» находятся документы «Document.cs», «Partner.cs», «Resource.cs», «Request.cs», «Contact.cs», «FavoriteResource.cs» и «project.assets.json». «Models» представляет собой модели. Модели – это данные и правила, которые используются для работы с данными, которые представляют концепцию управления приложением. В любом приложении вся структура моделируется как данные, которые обрабатываются определённым образом.

### 3.2 Разработка клиентской части приложения

Клиентская часть приложения – это скрипты, написанные на языке программирования Javascript (JS) и исполняемые в браузере пользователя. Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Клиентская часть Web-приложения разработана при помощи библиотеки Blazor. На рисунке 8 представлен скриншот из редактора кода «Visual Studio», где представлен каталог «ExtResBlzr» в котором расположены такие каталоги как:

- «bin»;
- «Controllers»;
- «Data»;
- «obj»;
- «Pages»;
- «Properties»;
- «Shared»;
- «wwwroot».

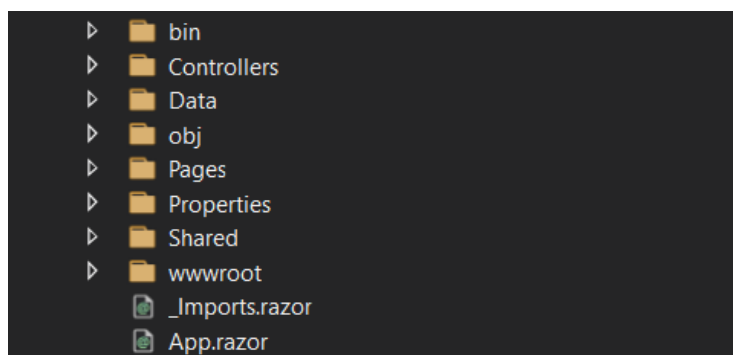


Рисунок 8

Папка bin содержит двоичные файлы, которые являются фактическим исполняемым кодом приложения.

Каталог «Controllers» получает запрос и система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки.

Каталог «Data» хранит в себе все данные из базы данных.

Папка obj содержит объектные или промежуточные файлы, которые представляют собой скомпилированные двоичные файлы, которые еще не были связаны. По сути, это фрагменты, которые будут объединены для создания окончательного исполняемого файла. Компилятор генерирует один объектный файл для каждого исходного файла, и эти файлы помещаются в папку obj .

Каталог «Pages» хранит в себе все страницы приложения.

Папка «Properties» создается автоматически и хранит в себе всю информацию о приложении.

В папке «Shared» хранятся компоновки и представления, не являющиеся специфичными для какого-либо контроллера.

Папка «wwwroot» является корнем приложения.

### 3.3 Описание используемых функций и процедур

При разработке Web-приложения используется фреймворк Blazor, предназначенный для создания интерактивных приложений, которые могут работать как на стороне сервера, так и на стороне клиента, на платформе .NET. Компоненты аналогичны функциям C#. Здесь компонент представляет элемент интерфейса, например, какое-то определенное содержание, меню, диалоговое окно, форма ввода данных. Компоненты определяют логику рендеринга элементов интерфейса, а также логику обработки пользовательского ввода. Компоненты могут быть вложенными в другие компоненты. Компоненты можно повторно использовать в других проектах и переносить в виде библиотеки классов Razor.

Все компоненты находятся в папке «Models».

Компонент «AddResource» находится в документе «ResourceRepository.cs» код которого представлен ниже, отвечает за создание резюме.

```
public async Task<Resource> AddResource(Resource Resource)
{
    //generate unique id
    Guid guid = Guid.NewGuid();
    Resource.Id = guid;

    var result = await _appDbContext.Resource.AddAsync(Resource);

    await _appDbContext.SaveChangesAsync();
    return result.Entity;
}

@page "/createResource"
@Inject IResourceRepository Service
@Inject IPartnerRepository partnersService
@Inject IResourceLevelRepository levService
```



```

@Inject NavigationManager navManger
<div>
    <h3>Create Resource</h3>

    <EditForm Model="@resource" OnSubmit="@HandleValidSubmit">
    @*<DataAnnotationsValidator />
    <ValidationSummary />*@
    <label>FirstName</label>
    <InputText id="firstName" @bind-Value="resource.FirstName" />
    <br />
    <label>LastName</label>
    <InputText id="lastName" @bind-Value="resource.LastName" />
    <br />
    <label>Partner</label>

    <select name="partnerId">
    <option value=""></option>
    @if (Partners != null)
    {
        foreach (var item in Partners)
        {
            <option value="@item.Id">@item.Name</option>
        }
    }
    </select>
    <br />
    <label>Title</label>
    <InputText id="title" @bind-Value="resource.Title" />
    <br />
    <label>About</label>
    <InputText id="about" @bind-Value="resource.About" />

    <label>Technologies</label>
    <InputText id="technology" @bind-Value="resource.Technologies" />

<label>Level Declared</label>

    <select name="level">
    <option value=""></option>
    @*@if (ResourceLevels != null)
    {
        foreach (var item in ResourceLevels)
        {
            <option value="@item.Id">@item.Name</option>
        }
    }*@
    </select>

    <br />

```

```

<label>English Spoken</label>
<input type="checkbox" id="enspoken" @bind-value="resource.EnglishSpoken" />
<br />

<label>English Feedback</label>
<InputText id="enfeed" @bind-Value="resource.EnglishFeedback" />

<label>Available</label>
<input type="checkbox" id="available" @bind-value="resource.Available" />
<br />
<label>Cvtool</label>
<InputText id="cv" @bind-Value="resource.CvtoolLinkMaster" />
<br />
<label>Added</label>
<InputDate id="added" @bind-Value="resource.Added" />

<label>Updated</label>
<InputDate id="updated" @bind-Value="resource.Updated" />

<button type="submit">Add</button>
</EditForm>

</div>
@code {
    private Resource resource = new Resource();
    private IEnumerable<DataModel.Partner> Partners;
    private IEnumerable<DataModel.ResourceLevel> ResourceLevels;

    private void HandleValidSubmit()
    {
        Service.AddResource(resource);
    }

    protected override async Task OnInitializedAsync()
    {
        Partners = await partnersService.GetPartners();
        //ResourceLevels = await levService.GetLevels();
    }

    // public List<int> p = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    public Partner partner = new Partner();
}
Компонент «ResourceList» находится в документе «ResourceList.razor» код которого
представлен ниже, отвечает за отображение сотрудников.

@page "/Resources"
@Inject DataModel.IResourceRepository Service

```

```

<div>
  <h3>Resource List</h3>
  <div class="form-group">
    <a class="btn btn-success" href="createResource">
<i class="oi oi-plus"></i> Create New</a>
  </div>
  <br>
</div>
<div>
  @if (Resources != null)
  {
    <table class="table">
      <thead>
        <tr>
          <th scope="col">ID</th>
          <th scope="col">FirstName</th>
          <th scope="col">LastName</th>
          <th scope="col">PartnerId</th>
          <th scope="col">Title</th>
          <th scope="col">About</th>
          <th scope="col">Technologies</th>
          <th scope="col">LevelDeclared</th>
          <th scope="col">EnglishSpoken</th>
          <th scope="col">EnglishFeedback</th>
          <th scope="col">Available</th>
          <th scope="col">CVToolLinkMaster</th>
          <th scope="col">Added</th>
          <th scope="col">Updated</th>

        </tr>
      </thead>
      <tbody>
        @foreach (var Resource in Resources)
        {
          <tr>
            <th>@Resource.Id</th>
            <td>@Resource.FirstName</td>
            <td>@Resource.LastName</td>
            <td>@Resource.PartnerId</td>
            <td>@Resource.Title</td>
            <td>@Resource.About</td>
            <td>@Resource.Technologies</td>
            <td>@Resource.LevelDeclared</td>
            <td>@Resource.EnglishSpoken</td>
            <td>@Resource.EnglishFeedback</td>
            <td>@Resource.Available</td>
            <td>@Resource.CvtoolLinkMaster</td>
            <td>@Resource.Added</td>
            <td>@Resource.Updated</td>

```

```

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

```

```

@code { private IEnumerable<DataModel.Resource> Resources;

protected override async Task OnInitializedAsync()
{
    Resources = await Service.GetResources();
}
}

```

Компонент «ContactRepository» находится в документе «ContactRepository.cs» код которого представлен ниже.

```

namespace DataModel
{
    public class ContactRepository : IContactRepository
    {
        private readonly PartnersContext _appDbContext;
        public ContactRepository(PartnersContext appDbContext)
        {
            _appDbContext = appDbContext;
        }

        public async Task<Contact> AddContact(Contact Contact)
        {
            var result = await _appDbContext.Contact.AddAsync(Contact);
            await _appDbContext.SaveChangesAsync();
            return result.Entity;
        }

        public async Task<Contact> DeleteContact(Guid id)
        {
            var Contact = await _appDbContext.Contact.FindAsync(id);
            if (Contact != null)
            {
                _appDbContext.Contact.Remove(Contact);
                await _appDbContext.SaveChangesAsync();
            }
            return Contact;
        }

        public async Task<Contact> GetContact(Guid ContactId)
        {
            var result = await _appDbContext.Contact
                .Include(e => e.Partner)

```

```

        .FirstOrDefaultAsync(e => e.Id == ContactId);
        return result;
    }

    public async Task<IEnumerable<Contact>> GetContacts()
    {
        return await _appDbContext.Contact.Include(e => e.Partner).ToListAsync();
    }

    public async Task<Contact> UpdateContact(Contact Contact)
    {
        var result = await _appDbContext.Contact.FirstOrDefaultAsync(e => e.Id == Contact.Id);

        if (result != null)
        {
            result.FirstName = Contact.FirstName;
            result.LastName = Contact.LastName;
            //TODO

            await _appDbContext.SaveChangesAsync();
        }
        return result;
    }
}

namespace DataModel
{
    public interface IContactRepository
    {
        Task<IEnumerable<Contact>> GetContacts();
        Task<Contact> GetContact(Guid ContactId);
        Task<Contact> AddContact(Contact Contact);
        Task<Contact> UpdateContact(Contact Contact);
        Task<Contact> DeleteContact(Guid ContactId);
    }
}

```

Полный код web-приложения представлен в приложении А.

## 4 Описание Web-приложения

### 4.1 Общие сведения

Главная цель – достичь максимального качества выполненных задач, предоставить готовый, высокотехнологичный продукт, соответствующий современным требованиям. Сделать удобным, доступным сервис по обслуживанию и дальнейшему содержанию сайта.

Для установки приложения необходимо скачать и переместить папку с проектом в удобное для запуска место. Запуск приложения осуществляется при помощи клавиш `fn + f5`. Верстка страниц осуществляется при помощи фреймворка Blazor.

### 4.2 Функциональное назначение

Web-приложение для подбора персонала на проекты направлено на предоставление возможности поиска необходимых людей для создания проектов, возможности удобного просмотра всех людей, возможности загрузки резюме сотрудника в базу.

Потенциальной аудиторией являются менеджеры проектов, которые могут сохранить свое время на заполнении необходимой информации о проектах, сотрудниках и заказчиках посредством использования разработанного приложения.

Для каждого пользователя существует отдельный аккаунт, что позволяет пользователям управлять только данными их компании и защитить от нежелательных изменений информации от других сотрудников. Все пароли, хранящиеся в базе данных строго зашифрованы, что добавляет безопасности при случае взлома.

Данное приложение поддерживает все популярные браузеры, включая Internet Explorer 9 и выше.

### 4.3 Описание разделов сайта

Сайт имеет навигационную панель, которая находится в верхней шапке и отображается на каждой странице. С помощью навигационной панели можно перейти на страницу своего профиля, где отображается вся информация профиля, на страницу подписок, где отображаются все записи пользователей, на который подписан авторизованный пользователь, на вкладку добавление поста, для его добавления соответственно, а также выйти из своего профиля нажав на кнопку выйти.

Так же имеются страница авторизации и регистрации. При успешной регистрации мы получаем уведомление о том, что регистрация прошла успешно и можно войти. При ошибках входа и регистрации появляется уведомление об ошибке.

Главная страница отображает всех сотрудников в организации, заказчиков, резюме и текущие проекты.

## 5 Методика испытаний

### 5.1 Технические требования

Минимальные системные требования для проведения технических испытаний:

- 400 МБ свободного места на жестком диске;
- 1 ГБ оперативной памяти;
- процессор Intel Core i5 1,6 ГГц;
- операционная система семейства Microsoft Windows 10;
- Microsoft .NET Framework 4.5.2;
- интернет 2048/512 (Кбит/с).

Все технические испытания будут проводиться в браузере Google Chrome версии 81.0.4044.122 на стандартных настройках.

Компьютер должен работать под управлением операционной системы Windows 10. На компьютере должен быть установлен браузер. Браузер – это специальная программа, которая предназначена для просмотра Web-страниц. Рекомендуется использовать браузер Google Chrome.

В настоящее время существует много Web-браузеров, каждый из которых обладает своими преимуществами и недостатками. Одним из самых популярных и используемых на компьютере это браузеры – Internet Explorer, Mozilla Firefox, Opera, Safari, Google Chrome. И по большому счету нет большой разницы, какой из них используете.

### 5.2 Функциональное тестирование

В процессе написания приложения необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование web-приложения будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены. Тест кейсы представлены ниже в таблицах 11-13.

Таблица 11 – Тест-кейс для регистрации

№ тест-кейса	Модуль/Функция	Шаги воспроизведения	Результат
001	Регистрация	1. Запустить приложение; 2. В загруженном окне заполнить форму, пример представлен на рисунке 9;	Ожидаемый результат: Переход на страницу авторизации, отображение главной страницы приложения.

		<p>3. Заполнить поле «FirstName» значением «Kate»;</p> <p>4. Заполнить поле «LastName» значением «Vuhouskaya»;</p> <p>5. Заполнить поле «Email» значением «qwerty@gmail.com»;</p> <p>6. Заполнить поле «Office Phone» значением «8800553535»;</p> <p>7. Заполнить поле «Postal Code» значением «222720»;</p> <p>8. Заполнить поле «City» значением «Минск»;</p> <p>9. Заполнить поле «Country» значением «Беларусь»;</p> <p>10. Нажать на кнопку «Next».</p>	<p>Фактический результат: Переход на страницу авторизации и отображение главной станицы приложения, пример которого представлен на рисунке 10.</p>
--	--	--	--

Рисунок 9

Рисунок 10



Таблица 12 - Тест-кейс для авторизации

№ тест-кейса	Модуль/ Функция	Шаги воспроизведения	Результат
002	Авторизация администратора	1. Запустить приложение; 2. Перейти к форме авторизации. Пример представлен на рисунке 11; 3. Заполнить поле «login» значением «admin»; 4. Заполнить поле «password» значением «admin»; 5. Нажать на кнопку «LOGIN».	Ожидаемый результат: Переход на главную страницу.
			Фактический результат: Переход на главную страницу Пример представлен на рисунке 12.



Рисунок 11

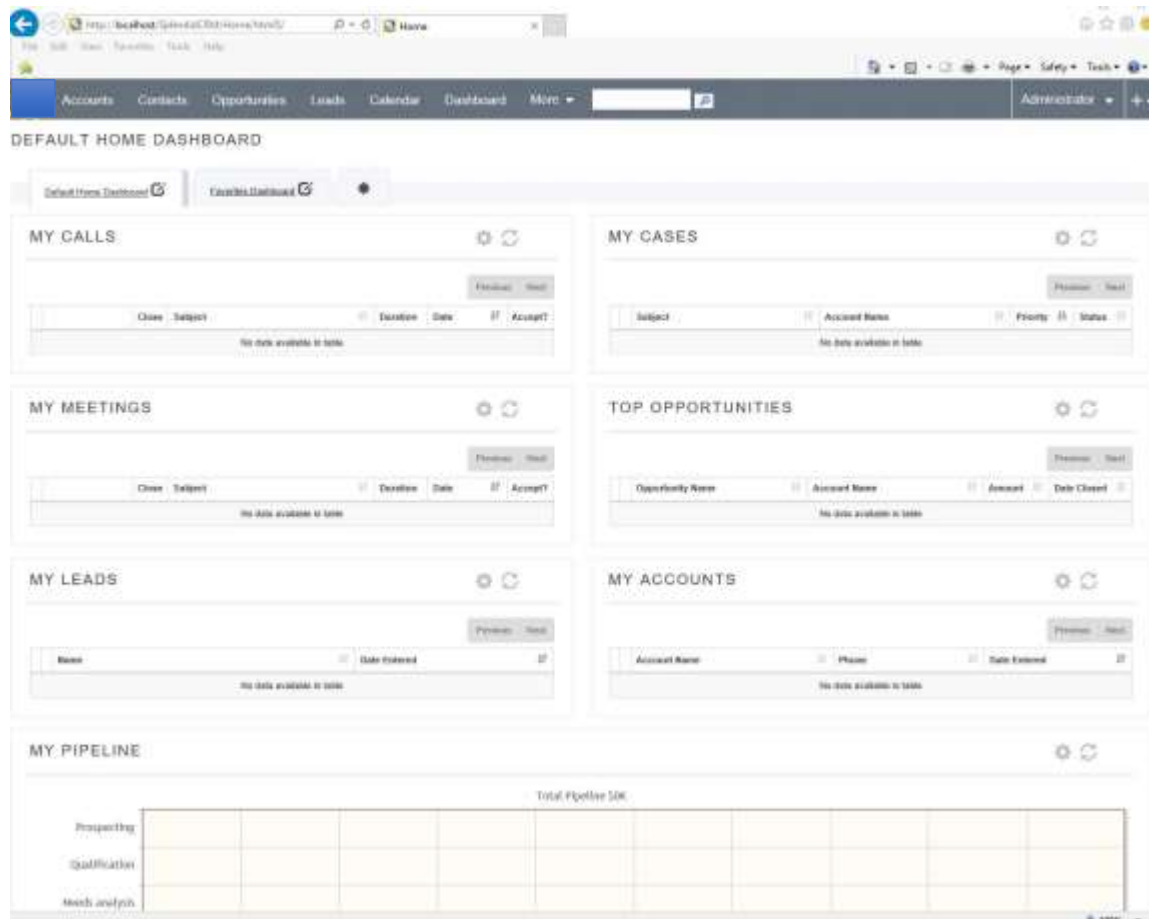


Рисунок 12

Таблица 13 - Тест-кейс для добавления партнера

№ тест-кейса	Модуль/Функция	Шаги воспроизведения	Результат
003	Добавление партнера	1. Запустить приложение; 2. Перейти на вкладку «Contacts»; 3. В выпадающем списке выбрать «Create Contact»; 4. Заполнить форму данными, представленную на рисунке 13; 5. Заполнить поле «FirstName» значением «Aaaa»; 6. Заполнить поле «LastName» значением «Bbb»; 7. Заполнить поле «Account Name» значением «Aba Abnovich»; 8. В поле «Lead Source» выбрать значение «Web site»; 9. Заполнить поле «Title» значением «Aaaaaaaaaaaaaaaaaaaaaa»; 10. Заполнить поле «Department» значением «Moscow»;	Ожидаемый результат: Добавление нового партнера.
			Фактический результат: Добавление партнера и отображение в списке всех партнеров, пример представлен на рисунке 14.

		11. Поставить галочку в поле «Do not Call»; 12. Нажать на кнопку «Save».	
--	--	---	--

## CONTACTS

SAVE

CANCEL

### CONTACT INFORMATION

First Name:

--None--

Last Name:\*

Account Name:

SELECT

CLEAR

Lead Source:

--None--

Title:

Department:

Birthdate:

(MM/DD/YYYY)

Reports To:

SELECT

CLEAR

Sync Contact:

☐

Do Not Call:

☒

Assigned To:

Kate Vyhouskaya

SELECT

CLEAR

SAVE

CANCEL

Рисунок 13

HOME

Basic Search | Advanced Search | Duplicate Search

First Name:  Last Name:  Account Name:  SELECT


My Items: ☐ My Favorites: ☐

SEARCH CLEAR | Saved Searches: None | Save Item

---

CONTACT LIST Filter List Clear All Export

Refresh 1/1 1/1

	Name	Title	Account Name	Email Address	Phone	Tags	Date Modified	Assigned User
<input type="checkbox"/>	 <b>Aba Abnovich</b>		Abnovich				04/16/2021	Kate Vynnycky

Search Page - Results: 1/1 - Showing: 1/1 | Results: 1

MASS UPDATE

Рисунок 14

## 6 Применение

### 6.1 Назначение приложения

Web-приложение для подбора персонала на проекты направлено на автоматизацию работы менеджера проекта, которое будет предоставлять возможность поиска необходимых людей для создания проектов, возможность удобного просмотра всех сотрудников, возможность загрузки резюме сотрудника в базу, а также удобное отображение.

Удобная реализация добавления сотрудника позволяет менеджеру проекта без проблем загрузить на сайт личные данные сотрудника.

### 6.2 Программно-аппаратное обеспечение сервера и клиента

Для корректной работы Web-приложения на компьютере пользователя должен быть установлен любой современный браузер.

Хостинг должен соответствовать следующим минимальным требованиям:

- предоставлять не менее 300Мб дискового пространства;
- поддерживать язык программирования C#;
- CPU не менее 100 МГц;
- выделять не менее 60 Гбайт трафика.

Хостинг должен быстро и оперативно реагировать на любые обращения, в решении проблемных вопросов используя e-mail, телефон, Telegram, Skype, WEB форму.

## Заключение

В рамках преддипломной практики было разработано web-приложение для подбора персонала на проекты «CRMforPM».

Для достижения цели преддипломной были решены следующие задачи:

- авторизация менеджера;
- добавление нового сотрудника в базу;
- редактирование информации о новом сотруднике;
- загрузка и отображение резюме сотрудников;
- просмотр доступных сотрудников для проектов.
- обеспечено хранение необходимой информации в базе данных;
- разработаны необходимые программные модули компонентов сайта для организации поиска информации;
- приложение корректно отображается на различных браузерах.

Web-приложение имеет интуитивно понятный графический интерфейс, позволяющий с минимальным знанием интернета пользоваться данным приложением.

Программа реализована в полном объёме и в соответствии с техническим заданием, полностью отложена и протестирована. Поставленные задачи были выполнены.

## Список информационных источников

- 1 Багласова Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 0101 «Программное обеспечение технологий» / Т.Г.Багласова. – Минск: КБП, 2020
- 2 Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2020
- 3 ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам. – М.: Из-во стандартов, 1995
- 4 Графический редактор UML диаграмм [Электронный ресурс]. – Appdiagrams, 2020. – Режим доступа : <https://app.diagrams.net/>. – Дата доступа : 05.04.2021
- 5 Windows 10 [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://microsoft.com/ru-ru/windows/get-windows-10/>. – Дата доступа 07.04.2021
- 6 Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://visualstudio.microsoft.com/ru/vs/>. – Дата доступа 02.04.2020.
- 7 Visual Studio Code [Электронный ресурс]. – VisualStudio, 2021. – Режим доступа : <https://code.visualstudio.com/>. – Дата доступа 07.04.2021.
- 8 Microsoft SQL Server [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://www.microsoft.com/en-us/sql-server/sql-server-downloads/>. – Дата доступа 10.04.2021.
- 9 Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/>. – Дата доступа: 05.04.2021
- 10 Фреймворк Blazor [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://metanit.com/sharp/blazor/>. – Дата доступа 02.04.2021.
- 11 Google Chrome [Электронный ресурс]. – Google, 2021. – Режим доступа : <https://www.google.com/intl/ru/chrome/>. – Дата доступа 02.04.2021.

Приложение А  
(обязательное)  
Текст модулей приложения

```
using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

//создание модели бд
namespace DataModel
{
    public partial class Contact
    {
        public Contact()
        {
            EngagementOfOwn = new HashSet<EngagementOfOwn>();
        }

        public Guid Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Position { get; set; }
        public string Skype { get; set; }
        public string Telegram { get; set; }
        public string Phone { get; set; }
        public string Email { get; set; }
        public string Notes { get; set; }
        public Guid? PartnerId { get; set; }
        public DateTime? DateOfBirth { get; set; }
        public int? PreferredWayOfCommunication { get; set; }
        public string Gender { get; set; }

        public virtual Partner Partner { get; set; }
        public virtual WayOfCommunication PreferredWayOfCommunicationNavigation { get; set; }
        [JsonIgnore]
        public virtual ICollection<EngagementOfOwn> EngagementOfOwn { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace DataModel
{
    //создание модели бд

    public partial class ContactsPartners
```



```

    {
        public Guid? PartnerId { get; set; }
        public Guid Id { get; set; }
        public Guid Expr1 { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Position { get; set; }
        public string Skype { get; set; }
        public string Phone { get; set; }
        public string Email { get; set; }
        public string Notes { get; set; }
        public string Name { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace DataModel
{
    public partial class Document
    {
        //создание модели

        public Document()
        {
            PartnerDocument = new HashSet<PartnerDocument>();
            ResourceDocument = new HashSet<ResourceDocument>();
        }

        public Guid Id { get; set; }
        public string FileName { get; set; }
        public DateTime FileDate { get; set; }
        public int FileSize { get; set; }
        public string ContentType { get; set; }
        public byte[] ContentData { get; set; }
        public string Notes { get; set; }

        [JsonIgnore]
        public virtual ICollection<PartnerDocument> PartnerDocument { get; set; }
        [JsonIgnore]
        public virtual ICollection<ResourceDocument> ResourceDocument { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

```

```

namespace DataModel
{
//создание модели бд

    public partial class EngagementOfOwn
    {
        public Guid Id { get; set; }
        public Guid ResourceWithRateId { get; set; }
        public Guid PartnerId { get; set; }
        public string ProjectCode { get; set; }
        public string ProjectJira { get; set; }
        public Guid PartnerManagerId { get; set; }
        public DateTime DateStart { get; set; }
        public DateTime? DateFinish { get; set; }
        public string Notes { get; set; }
        public int? PaymentId { get; set; }

        [JsonIgnore]
        public virtual Partner Partner { get; set; }
        [JsonIgnore]
        public virtual Contact PartnerManager { get; set; }
        public virtual PaymentWay Payment { get; set; }
        [JsonIgnore]
        public virtual ResourceWithRate ResourceWithRate { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

```

```

namespace DataModel
{
//создание модели бд

    public partial class EngagementOfPartner
    {
        public Guid Id { get; set; }
        public Guid ResourceWithRateId { get; set; }
        public string ProjectCode { get; set; }
        public string ProjectJira { get; set; }
        public string AccountManagerId { get; set; }
        public string ProjectManagerId { get; set; }
        public DateTime DateStart { get; set; }
        public DateTime? DateFinish { get; set; }
        public string Notes { get; set; }
        public int? PaymentId { get; set; }

        public virtual PaymentWay Payment { get; set; }
        [JsonIgnore]
    }
}

```

```

        public virtual ResourceWithRate ResourceWithRate { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Xml.Serialization;

namespace DataModel
{
    //создание модели бд

    public partial class FavoriteResource
    {
        public Guid ResourceId { get; set; }
        public string UserId { get; set; }

        public virtual Resource Resource { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using DataModel;

namespace DataModel
{
    //создание интерфейса для реализации

    public interface IContactRepository
    {
        Task<IEnumerable<Contact>> GetContacts();
        Task<Contact> GetContact(Guid ContactId);
        Task<Contact> AddContact(Contact Contact);
        Task<Contact> UpdateContact(Contact Contact);
        Task<Contact> DeleteContact(Guid ContactId);
    }
}

using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace DataModel
{
    public partial class NonSolicitationType
    {
        //создание модели бд

        public NonSolicitationType()

```

```

    {
        Partner = new HashSet<Partner>();
    }

    public int Id { get; set; }
    public string Name { get; set; }

    [JsonIgnore]
    public virtual ICollection<Partner> Partner { get; set; }
}
}
using System;
using System.Collections.Generic;
using System.Xml.Serialization;

namespace DataModel
{
    //создание модели бд

    public partial class PartnersContacts
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
        public Guid? PrimaryContact { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Position { get; set; }
        public int? Rating { get; set; }
        public string Description { get; set; }
        public string Technologies { get; set; }
        public string Location { get; set; }
        public string BillingAddress { get; set; }
        public DateTime? Ndate { get; set; }
        public int? Ndayears { get; set; }
        public bool? NdanonSolicitation { get; set; }
        public bool? NdanonSolicitationExtended { get; set; }
        public Guid Expr1 { get; set; }
        public string Expr2 { get; set; }
        public string Expr3 { get; set; }
        public Guid Expr4 { get; set; }
        public string Expr5 { get; set; }
        public string Expr6 { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using DataModel;
using Microsoft.EntityFrameworkCore;

```

```

namespace DataModel
{
//создание функций для работы с моделями

public class ContactRepository : IContactRepository
{
    private readonly PartnersContext _appDbContext;
    public ContactRepository(PartnersContext appDbContext)
    {
        _appDbContext = appDbContext;
    }

    public async Task<Contact> AddContact(Contact Contact)
    {
        var result = await _appDbContext.Contact.AddAsync(Contact);
        await _appDbContext.SaveChangesAsync();
        return result.Entity;
    }

    public async Task<Contact> DeleteContact(Guid id)
    {
        var Contact = await _appDbContext.Contact.FindAsync(id);
        if (Contact != null)
        {
            _appDbContext.Contact.Remove(Contact);
            await _appDbContext.SaveChangesAsync();
        }
        return Contact;
    }

    public async Task<Contact> GetContact(Guid ContactId)
    {
        var result = await _appDbContext.Contact
            .Include(e => e.Partner)
            .FirstOrDefaultAsync(e => e.Id == ContactId);
        return result;
    }

    public async Task<IEnumerable<Contact>> GetContacts()
    {
        return await _appDbContext.Contact.Include(e => e.Partner).ToListAsync();
    }

    public async Task<Contact> UpdateContact(Contact Contact)
    {
        var result = await _appDbContext.Contact.FirstOrDefaultAsync(e => e.Id == Contact.Id);

        if (result != null)
        {

```

```

        result.FirstName = Contact.FirstName;
        result.LastName = Contact.LastName;
        //TODO

        await _appDbContext.SaveChangesAsync();
    }
    return result;
}
}

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

namespace DataModel
{
    //создание интерфейса модели

    public interface IPartnerRepository
    {
        Task<IEnumerable<Partner>> GetPartners();
        Task<Partner> GetPartner(Guid PartnerId);
    }
}

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

namespace DataModel
{
    //создание интерфейса модели
    public interface IResourceLevelRepository
    {
        Task<IEnumerable<ResourceLevel>> GetLevels();
        Task<ResourceLevel> GetLevel(Guid RLevelId);
        Task<ResourceLevel> AddLevel(ResourceLevel RLevel);
        Task<ResourceLevel> UpdateLevel(ResourceLevel RLevel);
        Task<ResourceLevel> DeleteLevel(Guid RLevelId);
    }
}

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

namespace DataModel

```

```

{
//создание интерфейса для реализации
    public interface IResourceRepository
    {
        Task<IEnumerable<Resource>> GetResources();
        Task<Resource> GetResource(Guid ResourceId);
        Task<Resource> AddResource(Resource Resource);
        Task<Resource> UpdateResource(Resource Resource);
        Task<Resource> DeleteResource(Guid ResourceId);
    }
}
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using DataModel;
using Microsoft.EntityFrameworkCore;
using System.Linq;

namespace DataModel
{
//реализация интрфейса
    public class PartnerRepository : IPartnerRepository
    {
        private readonly PartnersContext _appDbContext;
        public PartnerRepository(PartnersContext appDbContext) => _appDbContext = appDbContext;

        public async Task<Partner> GetPartner(Guid PartnerId)
        {
            var result = await _appDbContext.Partner.FirstOrDefaultAsync(e => e.Id == PartnerId);
            return result;
        }

        public async Task<IEnumerable<Partner>> GetPartners()
        {
            return await _appDbContext.Partner.ToListAsync();
        }
    }
}
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataModel
{
    public class ResourceLevelRepository : IResourceLevelRepository
    {
        private readonly PartnersContext _appDbContext;

```

```

public ResourceLevelRepository(PartnersContext appDbContext)
{
    _appDbContext = appDbContext;
}
public async Task<ResourceLevel> AddLevel(ResourceLevel RLevel)
{
    var result = await _appDbContext.ResourceLevel.AddAsync(RLevel);

    await _appDbContext.SaveChangesAsync();
    return result.Entity;
}

public async Task<ResourceLevel> DeleteLevel(Guid RLevelId)
{
    var Level = await _appDbContext.ResourceLevel.FindAsync(RLevelId);
    if (Level != null)
    {
        _appDbContext.ResourceLevel.Remove(Level);
        await _appDbContext.SaveChangesAsync();
    }
    return Level;
}

public async Task<ResourceLevel> GetLevel(Guid RLevelId)
{
    throw new NotImplementedException();
}

public async Task<IEnumerable<ResourceLevel>> GetLevels()
{
    return await _appDbContext.ResourceLevel.Include(e =>
e.Name).ToListAsync();
}

public async Task<ResourceLevel> UpdateLevel(ResourceLevel RLevel)
{
    throw new NotImplementedException();
}
}

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataModel
{

```



```

public class ResourceRepository : IResourceRepository
{
    private readonly PartnersContext _appDbContext;
    public ResourceRepository(PartnersContext appDbContext)
    {
        _appDbContext = appDbContext;
    }

    public async Task<Resource> AddResource(Resource Resource)
    {
        //generate unique id
        Guid guid = Guid.NewGuid();
        Resource.Id = guid;

        var result = await _appDbContext.Resource.AddAsync(Resource);

        await _appDbContext.SaveChangesAsync();
        return result.Entity;
    }

    public async Task<Resource> DeleteResource(Guid ResourceId)
    {
        var Resource = await _appDbContext.Resource.FindAsync(ResourceId);
        if (Resource != null)
        {
            _appDbContext.Resource.Remove(Resource);
            await _appDbContext.SaveChangesAsync();
        }
        return Resource;
    }

    public async Task<Resource> GetResource(Guid ResourceId)
    {
        var result = await _appDbContext.Resource
            .Include(e => e.Partner)
            .FirstOrDefaultAsync(e => e.Id == ResourceId);
        return result;
    }

    public async Task<IEnumerable<Resource>> GetResources()
    {
        return await _appDbContext.Resource.Include(e => e.Partner).ToListAsync();
    }

    public async Task<Resource> UpdateResource(Resource Resource)
    {
        var result = await _appDbContext.Resource.FirstOrDefaultAsync(e => e.Id ==
Resource.Id);

        if (result != null)

```

```

        {
            result.FirstName = Resource.FirstName;
            result.LastName = Resource.LastName;
            //TODO

            await _appDbContext.SaveChangesAsync();
        }
        return result;
    }
}

@page "/Resources"
@Inject DataModel.IResourceRepository Service

<div>
    <h3>Resource List</h3>
    <div class="form-group">
        <a class="btn btn-success" href="createResource"><i class="oi oi-plus"></i> Create New</a>
    </div>
    <br>
</div>
<div>
    @if (Resources != null)
    {
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">ID</th>
                    <th scope="col">FirstName</th>
                    <th scope="col">LastName</th>
                    <th scope="col">ParnerId</th>
                    <th scope="col">Title</th>
                    <th scope="col">About</th>
                    <th scope="col">Technologies</th>
                    <th scope="col">LevelDeclared</th>
                    <th scope="col">EnglishSpoken</th>
                    <th scope="col">EnglishFeedback</th>
                    <th scope="col">Available</th>
                    <th scope="col">CVToolLinkMaster</th>
                    <th scope="col">Added</th>
                    <th scope="col">Updated</th>

                </tr>
            </thead>
            <tbody>
                @foreach (var Resource in Resources)
                {
                    <tr>
                        <th>@Resource.Id</th>

```

```

        <td>@Resource.FirstName</td>
        <td>@Resource.LastName</td>
        <td>@Resource.PartnerId</td>
        <td>@Resource.Title</td>
        <td>@Resource.About</td>
        <td>@Resource.Technologies</td>
        <td>@Resource.LevelDeclared</td>
        <td>@Resource.EnglishSpoken</td>
        <td>@Resource.EnglishFeedback</td>
        <td>@Resource.Available</td>
        <td>@Resource.CvtoolLinkMaster</td>
        <td>@Resource.Added</td>
        <td>@Resource.Updated</td>
    </tr>
}
</tbody>
</table>}
</div>

```

```

@code { private IEnumerable<DataModel.Resource> Resources;

    protected override async Task OnInitializedAsync()
    {
        Resources = await Service.GetResources();
    }
}

@page "/updateContact/{id}"
@Inject IContactRepository Service
@Inject NavigationManager navManager
<h3>UpdateContact</h3>
@page "/createResource"
@Inject IResourceRepository Service
@Inject IPartnerRepository partnersService
@Inject IResourceLevelRepository levService
@Inject NavigationManager navManger
<div>
    <h3>Create Resource</h3>

    <EditForm Model="@resource" OnSubmit="@HandleValidSubmit">
        @*<DataAnnotationsValidator />
        <ValidationSummary />*@
        <label>FirstName</label>
        <InputText id="firstName" @bind-Value="resource.FirstName" />
        <br />
        <label>LastName</label>
        <InputText id="lastName" @bind-Value="resource.LastName" />
        <br />
        <label>Partner</label>

        <select name="partnerId">

```

```

<option value=""></option>
@if (Partners != null)
{
    foreach (var item in Partners)
    {
        <option value="@item.Id">@item.Name</option>
    }
}
</select>
<br />
<label>Title</label>
<InputText id="title" @bind-Value="resource.Title" />
<br />
<label>About</label>
<InputText id="about" @bind-Value="resource.About" />

<label>Technologies</label>
<InputText id="technology" @bind-Value="resource.Technologies" />

<label>Level Declared</label>

<select name="level">
    <option value=""></option>
    @*@if (ResourceLevels != null)
    {
        foreach (var item in ResourceLevels)
        {
            <option value="@item.Id">@item.Name</option>
        }
    }*@
</select>

<br />
<label>English Spoken</label>
<input type="checkbox" id="enspoken" @bind-value="resource.EnglishSpoken" />
<br />

<label>English Feedback</label>
<InputText id="enfeed" @bind-Value="resource.EnglishFeedback" />

<label>Available</label>
<input type="checkbox" id="available" @bind-value="resource.Available" />
<br />
<label>Cvtool</label>
<InputText id="cv" @bind-Value="resource.CvtoolLinkMaster" />
<br />
<label>Added</label>
<InputDate id="added" @bind-Value="resource.Added" />

```

```

        <label>Updated</label>
        <InputDate id="updated" @bind-Value="resource.Updated" />

        <button type="submit">Add</button>
    </EditForm>

</div>
@code {
    private Resource resource = new Resource();
    private IEnumerable<DataModel.Partner> Partners;
    private IEnumerable<DataModel.ResourceLevel> ResourceLevels;

    private void HandleValidSubmit()
    {
        Service.AddResource(resource);
    }

    protected override async Task OnInitializedAsync()
    {
        Partners = await partnersService.GetPartners();
        //ResourceLevels = await levService.GetLevels();
    }

    // public List<int> p = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    public Partner partner = new Partner();

}
@page "/Contactdetails/{id}"
@inject IContactRepository Service
@inject NavigationManager navManger
<h3>ContactDetails</h3>

<div class="row justify content center m.3">
    <div class="col-sm-8">
        <div class="card">
            @if (Contact != null)
            {
                <div class="card-header">
                    <h1>@Contact.FirstName</h1>
                </div>
                <div class="card-body text-center">
                    <h4>Contact ID : @Contact.Id</h4>
                </div>
                <div class="card-footer text-center">
                    <a class="btn btn-primary" href="/">Back</a>
                    <a class="btn btn-primary" href="updateContact/@Contact.Id">Edit</a>
                    <button class="btn btn-danger" @onclick="DeleteContact">Delete</button>
                </div>
            }
        </div>
    </div>

```

</div>  
</div>

```
@code { [Parameter]
    public string Id { get; set; }

    public Contact Contact { get; set; } = new Contact();

    protected override async Task OnInitializedAsync()
    {
        Contact = await Service.GetContact(Guid.Parse(Id));
    }

    protected async void DeleteContact()
    {
        await Service.DeleteContact(Guid.Parse(Id));
        navManger.NavigateTo("/");
    } }

using DataModel;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace ExtResourcesBlazor
{
    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.
        // For more information on how to configure your application, visit
        https://go.microsoft.com/fwlink/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
            var cs = Configuration.GetConnectionString("DefaultConnection");
            services.AddDbContext<PartnersContext>(options => options.UseSqlServer(cs));

            services.AddRazorPages();
            services.AddServerSideBlazor();

            services.AddScoped<IPartnerRepository, PartnerRepository>();
            services.AddScoped<IResourceRepository, ResourceRepository>();
        }
    }
}
```

```

        services.AddScoped<IContactRepository, ContactRepository>();
        services.AddScoped<IResourceLevelRepository, ResourceLevelRepository>();
    }

    // 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();
        }
        else
        {
            app.UseExceptionHandler("/Error");
        }

        app.UseStaticFiles();

        app.UseRouting();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
            endpoints.MapBlazorHub();
            endpoints.MapFallbackToPage("/_Host");
        });
    }
}

using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ExtResourcesBlazor
{
    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.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using DataModel;

namespace ExtResourcesBlazor
{
    public class ContactsController : Controller
    {
        private readonly PartnersContext _context;

        public ContactsController(PartnersContext context)
        {
            _context = context;
        }

        // GET: Contacts
        public async Task<IActionResult> Index()
        {
            var partnersContext = _context.Contact.Include(c => c.Partner).Include(c =>
c.PreferredWayOfCommunicationNavigation);
            return View(await partnersContext.ToListAsync());
        }

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

            var contact = await _context.Contact
                .Include(c => c.Partner)
                .Include(c => c.PreferredWayOfCommunicationNavigation)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (contact == null)
            {
                return NotFound();
            }
        }
    }
}

```



```

        return View(contact);
    }

    // GET: Contacts/Create
    public IActionResult Create()
    {
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name");
        ViewData["PreferredWayOfCommunication"] = new
SelectList(_context.WayOfCommunication, "Id", "Name");
        return View();
    }

    // POST: Contacts/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,FirstName,LastName,Position,Skype,Telegram,Phone,Email,Notes,PartnerId,DateOf
Birth,PreferredWayOfCommunication,Gender")] Contact contact)
    {
        if (ModelState.IsValid)
        {
            contact.Id = Guid.NewGuid();
            _context.Add(contact);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name", contact.PartnerId);
        ViewData["PreferredWayOfCommunication"] = new
SelectList(_context.WayOfCommunication, "Id", "Name", contact.PreferredWayOfCommunication);
        return View(contact);
    }

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

        var contact = await _context.Contact.FindAsync(id);
        if (contact == null)
        {
            return NotFound();
        }
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name", contact.PartnerId);

```

```

        ViewData["PreferredWayOfCommunication"] = new
        SelectList(_context.WayOfCommunication, "Id", "Name", contact.PreferredWayOfCommunication);
        return View(contact);
    }

    // POST: Contacts/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(Guid id,
    [Bind("Id,FirstName,LastName,Position,Skype,Telegram,Phone,Email,Notes,PartnerId,DateOfBirth,PreferredWayOfCommunication,Gender")] Contact contact)
    {
        if (id != contact.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(contact);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ContactExists(contact.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name", contact.PartnerId);
        ViewData["PreferredWayOfCommunication"] = new
        SelectList(_context.WayOfCommunication, "Id", "Name", contact.PreferredWayOfCommunication);
        return View(contact);
    }

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

```

```

        return NotFound();
    }

    var contact = await _context.Contact
        .Include(c => c.Partner)
        .Include(c => c.PreferredWayOfCommunicationNavigation)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (contact == null)
    {
        return NotFound();
    }

    return View(contact);
}

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

private bool ContactExists(Guid id)
{
    return _context.Contact.Any(e => e.Id == id);
}
}
}

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

namespace ExtResourcesBlazor
{
    public class ResourceController : Controller
    {
        private readonly PartnersContext _context;

        public ResourceController(PartnersContext context)
        {
            _context = context;
        }
    }
}

```

```

    }

    // GET: Resources
    public async Task<IActionResult> Index()
    {
        var resourcesContext = _context.Resource.Include(c => c.Partner).Include(c =>
c.LevelDeclaredNavigation);
        return View(await resourcesContext.ToListAsync());
    }

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

        var resource = await _context.Resource
            .Include(c => c.Partner)
            .Include(c => c.LevelDeclaredNavigation)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (resource == null)
        {
            return NotFound();
        }

        return View(resource);
    }

    // GET: Resources/Create
    public IActionResult Create()
    {
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name");
        ViewData["LevelDeclared"] = new SelectList(_context.ResourceLevel, "Id",
"Name");
        return View();
    }

    // POST: Resource/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,FirstName,LastName,PartnerId,Title,About,Technologies,LevelDeclared,EnglishSpo
ken,EnglishFeedback,Available,CVToolLinkMaster,Added,Updated")] Resource resource)
    {
        if (ModelState.IsValid)

```

```

        {
            resource.Id = Guid.NewGuid();
            _context.Add(resource);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name",
resource.PartnerId);
        ViewData["LevelDeclared"] = new SelectList(_context.WayOfCommunication,
"Id", "Name", resource.LevelDeclared);
        return View(resource);
    }

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

        var resource = await _context.Resource.FindAsync(id);
        if (resource == null)
        {
            return NotFound();
        }
        ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name",
resource.PartnerId);
        ViewData["LevelDeclared"] = new SelectList(_context.WayOfCommunication,
"Id", "Name", resource.LevelDeclared);
        return View(resource);
    }

    // POST: Resources/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(Guid id, [Bind("Id, FirstName, LastName,
PartnerId, Title, About, Technologies, LevelDeclared, EnglishSpoken, EnglishFeedback, Available,
CVToolLinkMaster, Added, Updated")] Resource resource)
    {
        if (id != resource.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {

```

```

        try
        {
            _context.Update(resource);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ResourceExists(resource.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["PartnerId"] = new SelectList(_context.Partner, "Id", "Name",
resource.PartnerId);
    ViewData["LevelDeclared"] = new SelectList(_context.WayOfCommunication,
    "Id", "Name", resource.LevelDeclared);
    return View(resource);
}

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

    var res = await _context.Resource
        .Include(c => c.Partner)
        .Include(c => c.LevelDeclaredNavigation)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (res == null)
    {
        return NotFound();
    }

    return View(res);
}

// POST: Resource/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(Guid id)
{

```

```

        var res = await _context.Resource.FindAsync(id);
        _context.Resource.Remove(res);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool ResourceExists(Guid id)
    {
        return _context.Resource.Any(e => e.Id == id);
    }
}

}

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

namespace DataModel
{
    public partial class PartnersContext : DbContext
    {
        public PartnersContext()
        {
        }

        public PartnersContext(DbContextOptions<PartnersContext> options)
            : base(options)
        {
        }

        public virtual DbSet<Contact> Contact { get; set; }
        public virtual DbSet<ContactsPartners> ContactsPartners { get; set; }
        public virtual DbSet<Document> Document { get; set; }
        public virtual DbSet<EngagementOfOwn> EngagementOfOwn { get; set; }
        public virtual DbSet<EngagementOfPartner> EngagementOfPartner { get; set; }
        public virtual DbSet<FavoriteResource> FavoriteResource { get; set; }
        public virtual DbSet<NonSolicitationType> NonSolicitationType { get; set; }
        public virtual DbSet<Partner> Partner { get; set; }
        public virtual DbSet<PartnerDocument> PartnerDocument { get; set; }
        public virtual DbSet<PartnerRating> PartnerRating { get; set; }
        public virtual DbSet<PartnersContacts> PartnersContacts { get; set; }
        public virtual DbSet<PaymentWay> PaymentWay { get; set; }
        public virtual DbSet<Request> Request { get; set; }
        public virtual DbSet<RequestResource> RequestResource { get; set; }
        public virtual DbSet<Resource> Resource { get; set; }
        public virtual DbSet<ResourceAvailability> ResourceAvailability { get; set; }
        public virtual DbSet<ResourceDocument> ResourceDocument { get; set; }
        public virtual DbSet<ResourceFeedback> ResourceFeedback { get; set; }
        public virtual DbSet<ResourceLevel> ResourceLevel { get; set; }
        public virtual DbSet<ResourceRates> ResourceRates { get; set; }
    }
}

```

```

public virtual DbSet<ResourceWithRate> ResourceWithRate { get; set; }
public virtual DbSet<User> User { get; set; }
public virtual DbSet<UserAccess> UserAccess { get; set; }
public virtual DbSet<UserSession> UserSession { get; set; }
public virtual DbSet<WayOfCommunication> WayOfCommunication { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
optionsBuilder.UseSqlServer("Server=.\SQLEXPRESS01;Database=Partners;Trusted_Conn
ection=True;MultipleActiveResultsets=true");
    }
}

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Contact>(entity =>
    {
        entity.Property(e => e.Id).ValueGeneratedNever();

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

        entity.Property(e => e.Email).HasMaxLength(50);

        entity.Property(e => e.FirstName)
            .IsRequired()
            .HasMaxLength(50);

        entity.Property(e => e.Gender)
            .HasMaxLength(1)
            .IsUnicode(false)
            .IsFixedLength()
            .HasComment("M, F, U");

        entity.Property(e => e.LastName).HasMaxLength(50);

        entity.Property(e => e.Phone).HasMaxLength(50);

        entity.Property(e => e.Position).HasMaxLength(50);

        entity.Property(e => e.Skype).HasMaxLength(50);

        entity.Property(e => e.Telegram).HasMaxLength(50);

        /*entity.HasOne(d => d.Partner)
            .WithMany(p => p.Contact)
            .HasForeignKey(d => d.PartnerId)
            .HasConstraintName("FK_Contacts_Partners");*/
    }
}

```



```

        entity.HasOne(d => d.PreferredWayOfCommunicationNavigation)
            .WithMany(p => p.Contact)
            .HasForeignKey(d => d.PreferredWayOfCommunication)
            .HasConstraintName("FK_Contacts_WaysOfCommunication");
    });

    modelBuilder.Entity<ContactsPartners>(entity =>
    {
        entity.HasNoKey();

        entity.ToView("ContactsPartners");

        entity.Property(e => e.Email).HasMaxLength(50);

        entity.Property(e => e.FirstName).HasMaxLength(50);

        entity.Property(e => e.LastName).HasMaxLength(50);

        entity.Property(e => e.Name)
            .IsRequired()
            .HasMaxLength(50);

        entity.Property(e => e.Phone).HasMaxLength(50);

        entity.Property(e => e.Position).HasMaxLength(50);

        entity.Property(e => e.Skype).HasMaxLength(50);
    });

    modelBuilder.Entity<Document>(entity =>
    {
        entity.Property(e => e.Id).ValueGeneratedNever();

        entity.Property(e => e.ContentData).IsRequired();

        entity.Property(e => e.ContentType)
            .HasMaxLength(50)
            .IsUnicode(false);

        entity.Property(e => e.FileDate).HasColumnType("datetime");

        entity.Property(e => e.FileName)
            .IsRequired()
            .HasMaxLength(50);
    });

    modelBuilder.Entity<EngagementOfOwn>(entity =>
    {
        entity.HasIndex(e => e.Id)

```

```

        .HasName("IX_EngagementsOfOwn");

entity.Property(e => e.Id).ValueGeneratedNever();

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

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

entity.Property(e => e.ProjectCode).HasMaxLength(50);

entity.Property(e => e.ProjectJira).HasMaxLength(50);

entity.HasOne(d => d.Partner)
    .WithMany(p => p.EngagementOfOwn)
    .HasForeignKey(d => d.PartnerId)
    .OnDelete(DeleteBehavior.ClientSetNull)
    .HasConstraintName("FK_EngagementsOfOwn_Partners");

/*entity.HasOne(d => d.PartnerManager)
    .WithMany(p => p.EngagementOfOwn)
    .HasForeignKey(d => d.PartnerManagerId)
    .OnDelete(DeleteBehavior.ClientSetNull)
    .HasConstraintName("FK_EngagementsOfOwn_Contacts");*/

entity.HasOne(d => d.Payment)
    .WithMany(p => p.EngagementOfOwn)
    .HasForeignKey(d => d.PaymentId)
    .HasConstraintName("FK_EngagementsOfOwn_PaymentWays");

entity.HasOne(d => d.ResourceWithRate)
    .WithMany(p => p.EngagementOfOwn)
    .HasForeignKey(d => d.ResourceWithRateId)
    .OnDelete(DeleteBehavior.ClientSetNull)
    .HasConstraintName("FK_EngagementsOfOwn_ResourceWithRates");
});

modelBuilder.Entity<EngagementOfPartner>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.AccountManagerId).HasMaxLength(50);

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

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

    entity.Property(e => e.ProjectCode).HasMaxLength(50);

    entity.Property(e => e.ProjectJira)
        .HasMaxLength(50)

```

```

        .IsUnicode(false);

entity.Property(e => e.ProjectManagerId).HasMaxLength(50);

entity.HasOne(d => d.Payment)
    .WithMany(p => p.EngagementOfPartner)
    .HasForeignKey(d => d.PaymentId)
    .HasConstraintName("FK_EngagementsOfPartners_PaymentWays");

entity.HasOne(d => d.ResourceWithRate)
    .WithMany(p => p.EngagementOfPartner)
    .HasForeignKey(d => d.ResourceWithRateId)
    .OnDelete(DeleteBehavior.ClientSetNull)
    .HasConstraintName("FK_EngagementsOfPartners_ResourceWithRates");
});

modelBuilder.Entity<FavoriteResource>(entity =>
{
    entity.HasNoKey();

    entity.HasIndex(e => e.ResourceId)
        .HasName("IX_FavoriteResources");

    entity.Property(e => e.UserId).IsRequired();

    entity.HasOne(d => d.Resource)
        .WithMany()
        .HasForeignKey(d => d.ResourceId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_FavoriteResources_Resources");
});

modelBuilder.Entity<NonSolicitationType>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Name)
        .IsRequired()
        .HasMaxLength(50);
});

modelBuilder.Entity<Partner>(entity =>
{
    entity.HasIndex(e => e.Id)
        .HasName("IX_Partners");

    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Name)
        .IsRequired()

```

```

        .HasMaxLength(50);

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

entity.HasOne(d => d.NonSolicitationNavigation)
    .WithMany(p => p.Partner)
    .HasForeignKey(d => d.NonSolicitation)
    .HasConstraintName("FK_Partners_NonSolicitationTypes");

entity.HasOne(d => d.RatingNavigation)
    .WithMany(p => p.Partner)
    .HasForeignKey(d => d.Rating)
    .HasConstraintName("FK_Partners_Rating");
});

modelBuilder.Entity<PartnerDocument>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.HasOne(d => d.Document)
        .WithMany(p => p.PartnerDocument)
        .HasForeignKey(d => d.DocumentId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_PartnerDocuments_Documents");

    entity.HasOne(d => d.Partner)
        .WithMany(p => p.PartnerDocument)
        .HasForeignKey(d => d.PartnerId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_PartnerDocuments_Partners");
});

modelBuilder.Entity<PartnerRating>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Name)
        .IsRequired()
        .HasMaxLength(50);
});

modelBuilder.Entity<PartnersContacts>(entity =>
{
    entity.HasNoKey();

    entity.ToView("PartnersContacts");

    entity.Property(e => e.Expr2).HasMaxLength(50);

    entity.Property(e => e.Expr3).HasMaxLength(50);

```

```

entity.Property(e => e.Expr5).HasMaxLength(50);

entity.Property(e => e.Expr6).HasMaxLength(50);

entity.Property(e => e.FirstName).HasMaxLength(50);

entity.Property(e => e.LastName).HasMaxLength(50);

entity.Property(e => e.Name)
    .IsRequired()
    .HasMaxLength(50);

entity.Property(e => e.Ndadata)
    .HasColumnName("NDADate")
    .HasColumnType("date");

entity.Property(e
e.NdanonSolicitation).HasColumnName("NDANonSolicitation"); =>

entity.Property(e
e.NdanonSolicitationExtended).HasColumnName("NDANonSolicitationExtended"); =>

entity.Property(e => e.Ndayears).HasColumnName("NDAYears");

entity.Property(e => e.Position).HasMaxLength(50);
});

modelBuilder.Entity<PaymentWay>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Notes)
        .IsRequired()
        .HasMaxLength(50);
});

modelBuilder.Entity<Request>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.ApproximateDuration).HasMaxLength(50);

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

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

    entity.Property(e => e.IssuerAccountId).HasMaxLength(50);

    entity.Property(e => e.LinkCollab).IsUnicode(false);

```

```

entity.Property(e => e.LinkJira).IsUnicode(false);

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

entity.HasOne(d => d.IssuerPartner)
    .WithMany(p => p.Request)
    .HasForeignKey(d => d.IssuerPartnerId)
    .HasConstraintName("FK_Requests_Partners");

entity.HasOne(d => d.ResourceLevelNavigation)
    .WithMany(p => p.Request)
    .HasForeignKey(d => d.ResourceLevel)
    .HasConstraintName("FK_Requests_ResourceLevels");
});

modelBuilder.Entity<RequestResource>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.HasOne(d => d.Request)
        .WithMany(p => p.RequestResource)
        .HasForeignKey(d => d.RequestId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_RequestResources_Requests");

    entity.HasOne(d => d.ResourceWithRate)
        .WithMany(p => p.RequestResource)
        .HasForeignKey(d => d.ResourceWithRateId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_RequestResources_Resources");

    entity.HasOne(d => d.ResourceWithRateNavigation)
        .WithMany(p => p.RequestResource)
        .HasForeignKey(d => d.ResourceWithRateId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_RequestResources_ResourceWithRates");
});

modelBuilder.Entity<Resource>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Added).HasColumnType("datetime");

    entity.Property(e
e.CvtoolLinkMaster).HasColumnName("CVToolLinkMaster");

    entity.Property(e => e.FirstName)
        .IsRequired()

```

=>

```

        .HasMaxLength(50);

entity.Property(e => e.LastName).HasMaxLength(50);

entity.Property(e => e.Updated).HasColumnType("datetime");

entity.HasOne(d => d.LevelDeclaredNavigation)
    .WithMany(p => p.Resource)
    .HasForeignKey(d => d.LevelDeclared)
    .HasConstraintName("FK_Resources_LevelDeclared");

entity.HasOne(d => d.Partner)
    .WithMany(p => p.Resource)
    .HasForeignKey(d => d.PartnerId)
    .HasConstraintName("FK_Resources_Partners");
});

modelBuilder.Entity<ResourceAvailability>(entity =>
{
    entity.HasIndex(e => e.AvailableFrom)
        .HasName("IX_ResourceAvailability");

    entity.Property(e => e.Id).ValueGeneratedNever();

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

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

    entity.Property(e => e.Notes)
        .HasMaxLength(10)
        .IsFixedLength();

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

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

    entity.HasOne(d => d.Resource)
        .WithMany(p => p.ResourceAvailability)
        .HasForeignKey(d => d.ResourceId)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_ResourceAvailability_Resources");
});

modelBuilder.Entity<ResourceDocument>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.HasOne(d => d.Document)
        .WithMany(p => p.ResourceDocument)
        .HasForeignKey(d => d.DocumentId)

```

```

        .OnDelete>DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_ResourceDocuments_Documents");

entity.HasOne(d => d.Resource)
    .WithMany(p => p.ResourceDocument)
    .HasForeignKey(d => d.ResourceId)
    .OnDelete>DeleteBehavior.ClientSetNull)
    .HasConstraintName("FK_ResourceDocuments_Resources");
});

modelBuilder.Entity<ResourceFeedback>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

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

    entity.Property(e => e.ScnSoftAccountId)
        .IsRequired()
        .HasMaxLength(50);

    entity.HasOne(d => d.LevelIdentifiedNavigation)
        .WithMany(p => p.ResourceFeedback)
        .HasForeignKey(d => d.LevelIdentified)
        .OnDelete>DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_ResourceFeedbacks_ResourceLevels");

    entity.HasOne(d => d.Resource)
        .WithMany(p => p.ResourceFeedback)
        .HasForeignKey(d => d.ResourceId)
        .OnDelete>DeleteBehavior.ClientSetNull)
        .HasConstraintName("FK_ResourceFeedbacks_Resources");
});

modelBuilder.Entity<ResourceLevel>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Name).HasMaxLength(50);
});

modelBuilder.Entity<ResourceRates>(entity =>
{
    entity.HasNoKey();

    entity.ToView("ResourceRates");

    entity.Property(e => e.Currency)
        .HasMaxLength(3)
        .IsUnicode(false)
        .IsFixedLength();
}

```



```

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

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

entity.Property(e => e.RateIn).HasColumnType("numeric(18, 0)");

entity.Property(e => e.RateOut).HasColumnType("numeric(18, 0)");
});

modelBuilder.Entity<ResourceWithRate>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

    entity.Property(e => e.Currency)
        .HasMaxLength(3)
        .IsUnicode(false)
        .IsFixedLength();

    entity.Property(e
e.CvtoolLinkSnapshot).HasColumnName("CVToolLinkSnapshot");

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

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

    entity.Property(e => e.RateIn).HasColumnType("numeric(18, 0)");

    entity.Property(e => e.RateOut).HasColumnType("numeric(18, 0)");

    entity.Property(e => e.ScnSoftResourceAccountId).HasMaxLength(50);

    entity.HasOne(d => d.PartnerResource)
        .WithMany(p => p.ResourceWithRate)
        .HasForeignKey(d => d.PartnerResourceId)
        .HasConstraintName("FK_ResourceWithRates_Resources");
});

modelBuilder.Entity<User>(entity =>
{
    entity.Property(e => e.Id).HasMaxLength(50);

    entity.Property(e => e.PersonalAppeal)
        .HasMaxLength(50)
        .IsUnicode(false);
});

modelBuilder.Entity<UserAccess>(entity =>
{
    entity.Property(e => e.Id).ValueGeneratedNever();

```

=>

```

        entity.Property(e => e.UserOrGroupId)
            .IsRequired()
            .HasMaxLength(50);
    });

    modelBuilder.Entity<UserSession>(entity =>
    {
        entity.HasNoKey();

        entity.HasIndex(e => e.Date)
            .HasName("IX_UserSession_1");

        entity.HasIndex(e => e.UserId)
            .HasName("IX_UserSession");

        entity.Property(e => e.Url).IsRequired();

        entity.Property(e => e.UserId)
            .IsRequired()
            .HasMaxLength(50);
    });

    modelBuilder.Entity<WayOfCommunication>(entity =>
    {
        entity.Property(e => e.Id).ValueGeneratedNever();

        entity.Property(e => e.Name)
            .IsRequired()
            .HasMaxLength(50);
    });

    OnModelCreatingPartial(modelBuilder);
}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
}

```