

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ФИЛИАЛ «РКТ» МАИ В Г. ХИМКИ МОСКОВСКОЙ ОБЛАСТИ

**ОТЧЁТ**

По дисциплине «Основы проектирования баз данных»

На тему:

«Разработка веб-приложения “Страховая компания”»

Специальность 09.02.07 — «Информационные системы и программирование»

Объем отчета 37 страниц

Выполнила:

студентка 2 курса

группы ИСП-21-19

Кузьмина А. Д.

Преподаватель:

Гуров В. В.

Химки, 2021

Оглавление

[Введение 3](#_Toc73204492)

[1.Создание базы данных 5](#_Toc73204493)

[1.1 ER диаграмма 5](#_Toc73204494)

[1.2 Реляционная схема 10](#_Toc73204495)

[1.3 SQL скрипт 12](#_Toc73204496)

[1.4 Генерация базы данных 13](#_Toc73204497)

[2. Создание приложения 17](#_Toc73204498)

[2.1 Создание проекта “Веб-приложение ASP.NET Core” 17](#_Toc73204499)

[2.2 Генерация контекста и моделей 23](#_Toc73204500)

[2.3 Миграция и переход на локальный сервер 24](#_Toc73204501)

[2.4. Создание веб-страниц с представлениями и действиями 26](#_Toc73204502)

[2.5 создание запросов и фильтров 30](#_Toc73204503)

[Заключение 36](#_Toc73204504)

# **Введение**

В современном мире объемы информации, необходимой людям для работы очень велики. Одной из удобнейших форм хранения информации, обеспечивающей также взаимодействие с ней, являются веб-сайты

Веб-сайты могут использоваться и используются во всех сферах деятельности. Сайты упрощают работу с информацией во много раз.

В данной работе будет рассмотрено создание веб-сайта с базой данных «Библиотека».

Наличие веб-сайта библиотеки упрощают работу сотрудников, а также полезны и для читателей. Реализация в цифровом виде картотек, списков книг, а также хранения информации об отделе кадров уменьшит затраты времени и сил для пользователей.   
Создаваться веб-сайт будет с помощью рабочей среды Visual Studio 2019 от компании Microsoft и их модуль для работы с базами данных Entity Framework. Оба продукта можно получить и использовать бесплатно. Также для обоих продуктов есть документация на официальном сайте Microsoft.

**В базе данных будут следующие таблицы:**

1. Сотрудники (Код сотрудника, ФИО, Возраст, Пол, Адрес, Телефон
2. Паспортные данные, Код должности)[10 записей].
3. Должности (Код должности, Наименование должности, Оклад, Обязанности, Требования)[5 записей].
4. Издательства (Код издательства, Наименование, Город, Адрес) [5 записей].
5. Жанры (Код жанра, Наименование, Описание) [5 записей].
6. Книги (Код книги, Наименование, Автор, Код издательства, Год издания, Код жанра) [10 записей].
7. Читатели (Код читателя, ФИО, Дата рождения, Пол, Адрес, Телефон, Паспортные данные) [10 записей].
8. Выданные книги (Код книги, Код читателя, Дата выдачи, Дата возврата, Отметка о возврате, Код сотрудника) [10 записей].

**Запросы:**

1. Отдел кадров (Связывает таблицы "Сотрудники" и "Должности" по полю "Код должности").
2. Каталог (Связывает таблицы "Книги", "Издательства" и "Жанры" по полям "Код издательства" и "Код жанра").
3. Книги на руках (Связывает таблицы "Выданные книги", "Книги", "Читатели" и "Сотрудники" по полям "Код книги", "Код читателя" и "Код сотрудника")

**Фильтры:**

1. Фильтры для отображения сотрудников отдельных должностей (На основе запроса "Отдел кадров").
2. Фильтры для отображения книг отдельных издательств (На основе запроса "Каталог").
3. Фильтры для отображения книг отдельных авторов (На основе запроса "Каталог").
4. Фильтры для отображения книг отдельных годов издания (На основе запроса "Каталог").
5. Фильтры для отображения сданных и не сданных книг (На основе запроса " Книги на руках ").
6. Фильтры для отображения книг на руках отдельных читателей (На основе запроса " Книги на руках ").

(Реализованные запросы и фильтры подчеркнуты)

# **Создание базы данных**

## **1.1 ER диаграмма**

Для создания исходного файла базы данных необходим SQL код, который будет сгенерирован на основе ER диаграммы, созданной с помощью онлайн-сервиса <https://erdplus.com>

Чтобы начать работать с ER-диаграммами, нужно во вкладке “DOCUMENTS” нажать на кнопку “NEW DIAGRAM…”, выбрать “ER Diagram”, написать имя диаграммы и нажать на кнопку “CREATE”.



Рис. 1.1 Вкладка “DOCUMENTS”

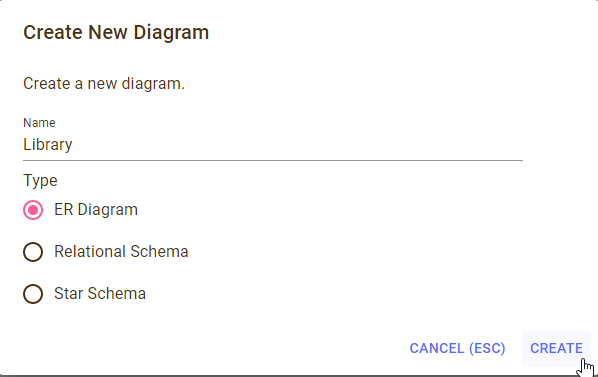


Рис. 1.2 Окно создание диаграммы “Create New Diagram”.

ER диаграмма состоит из сущностей, их атрибутов и связей. Сущности (Entity) в последствии будут преобразованы в таблицы базы данных, а атрибуты в поля этих таблиц.  
Располагаются сущности в рабочей области с помощью кнопки Entity, после чего сущности задается необходимое имя.



Рис. 1.3. кнопка добавления сущности

Атрибуты сущности создаются с помощью кнопки add attribute, после чего им также задаются имена.

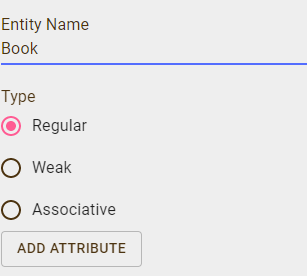


Рис. 1.4. добавление атрибута

Для атрибутов-первичных ключей следует указать, что они уникальны.

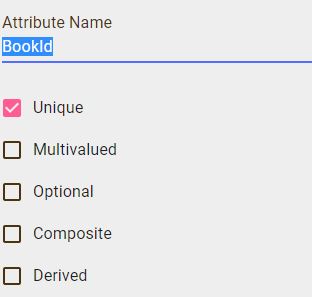


Рис. 1.5 указание уникальности поля (атрибута)

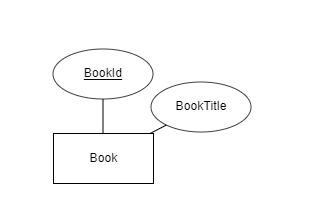


Рис. 1.6 сущность с атрибутами, уникальный атрибут подчеркнут

Чтобы связать сущности используется кнопка Connect.



Рис. 1.7. кнопка соединения

Для связи также можно задать имя. В настройках связи можно указать тип связи. Связь может быть “один к одному”, “один ко многим” и “многие ко многим”. Сущность в связи может быть *обязательной* или *необязательной* (отображается I или 0 соответсвенно)*.*

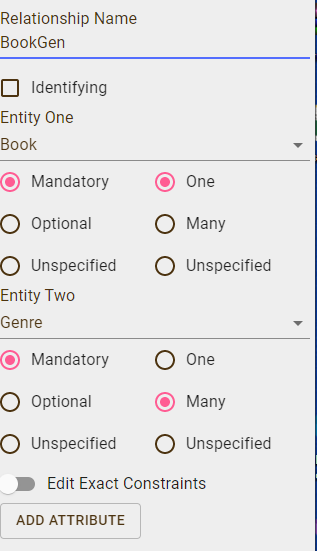


Рис. 1.8. Настройка связи между двумя сущностями.

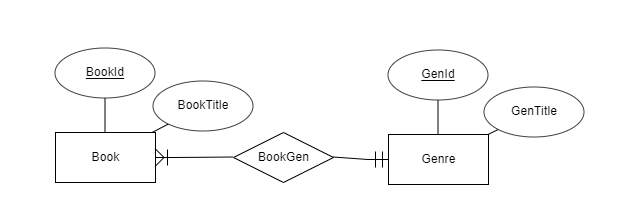


Рис. 1.9 связанные сущности

Готовая ER диаграмма для предметной области «библиотека»:

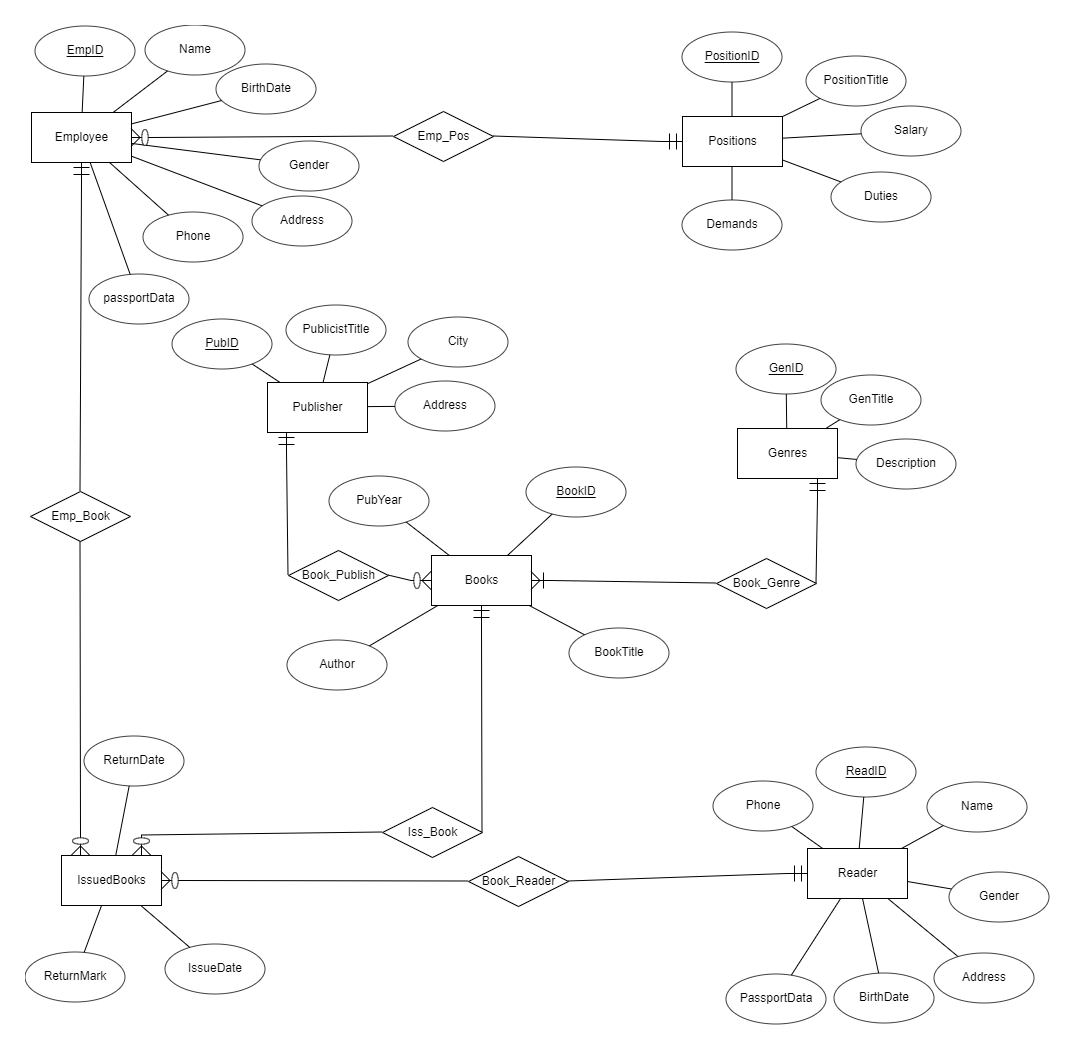


Рис. 1.10 Законченная ER-диаграмма.

На этом работа с ER-диаграммой заканчивается и начинается работа с реляционной схемой.

## **Реляционная схема**

Для создания реляционной схемы используя существующую ER-диаграмму, находясь во вкладке “DOCUMENTS”, нужно найти необходимую диаграмму и в правой части экрана напротив её названия нажать на 3 точки, выбрать пункт “Convert to Relational Schema”, выбрать название схемы и нажать на кнопку “CREATE&EDIT”.

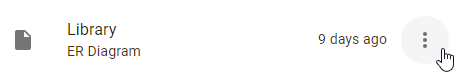


Рис. 1.11 Кнопка для открытия дополнительных действия над файлом.

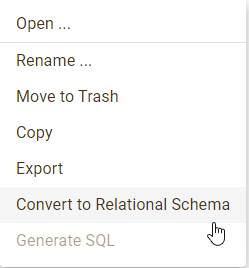


Рис. 1.12. Дополнительные действия над файлом ER-диаграммы.

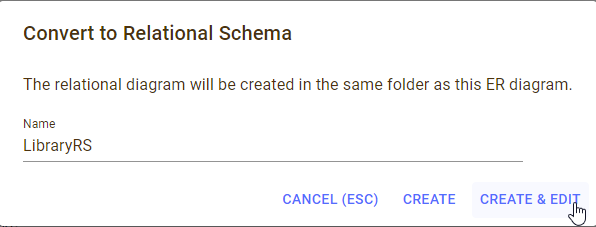


Рис. 1.13 преобразование ERдиаграммы в реляционную схему

Сгенерированная реляционная схема:

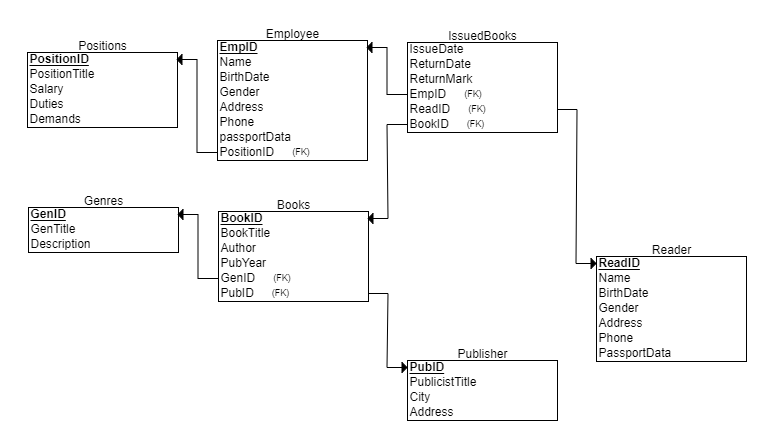


Рис. 1.14. Схема, полученная при преобразовании ER-диаграммы.

В качестве имени внешнего ключа используется имя главного ключа таблицы, с которая данная таблица связанна.

Также при работе с реляционной схемой у таблиц можно менять типы данных, нажав на сам тип напротив названия столбца.

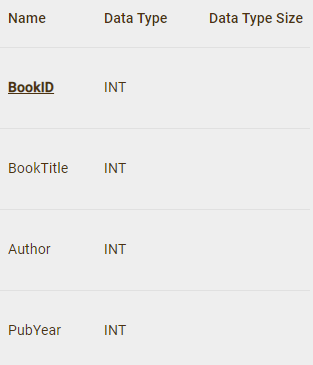


Рис. 1.15 Изменение типа данных у столбца таблицы.

На этом работа с реляционной схемой закончена.

## **1.3 SQL скрипт**

Чтобы получить SQL-скрипт из реляционной схемы, нужно, находясь во вкладке “DOCUMENTS”, нажать на 3 точки, находящиеся на против названия схемы, и выбрать пункт “Generate SQL”.

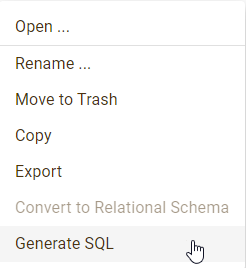


Рис. 1.16 Дополнительные действия над файлом реляционной схемы.

После чего необходимо создать файл с расширением .sql и вставить в него полученный sql скрипт.

Для создания базы данных мы будем использовать приложение SQLite Studio.

## **1.4 Генерация базы данных**

Для начала работы необходимо создать базу данных. Это можно сделать с помощью кнопки “Add a database”.

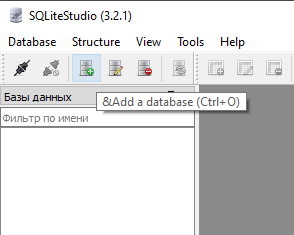


Рис. 1.17. Кнопка добавления новой базы данных “Add a database”.

Далее необходимо задать имя файла и выбрать его расположение.

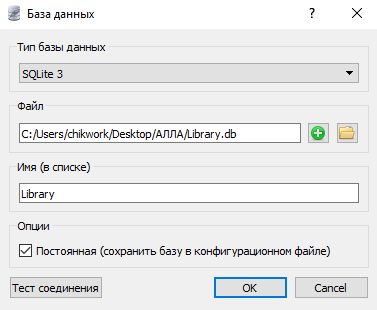


Рис. 1.18. Добавление новой базы данных.

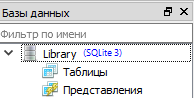


Рис. 1.19. Добавленная база данных.

Чтобы воспользоваться SQL-скриптом для создания базы данных, нужно нажать правой кнопкой по базе и выбрать пункт “Execute SQL from file”.

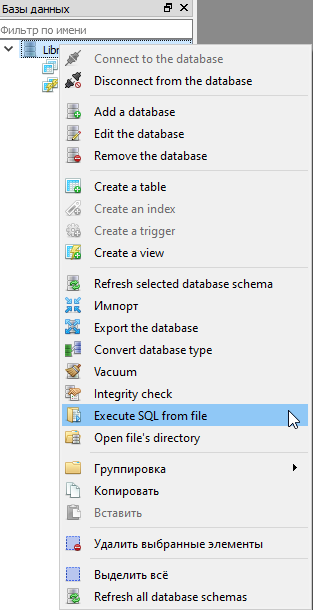


Рис. 1.20. Действия над базой данных.

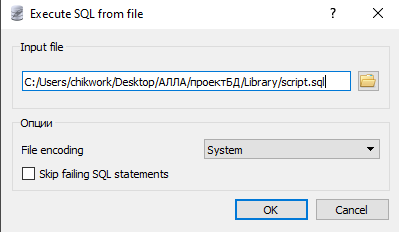
После чего необходимо выбрать путь к файлу, В котором сохранен скрипт.  


Рис. 1.21. Выбор файла для выполнения SQ-скрипта.

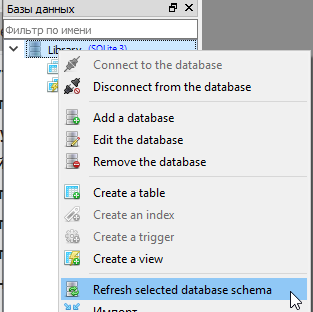
После этого нужно обновить базу данных, нажав на кнопку “Refresh selected database schema”.  


Рис. 1.22 Кнопка обновления баз данных “Refresh selected database schema”.

Некоторые типы данных, существующие в SQLiteStudio, не поддерживаются в Visual Studio (например, “varchar”), где мы будем продолжать работу с базой данных, поэтому лучше сразу открыть каждую таблицу и заменить типы данных на те, которые будут поддерживаться в Visual Studio, и сохранить изменения.

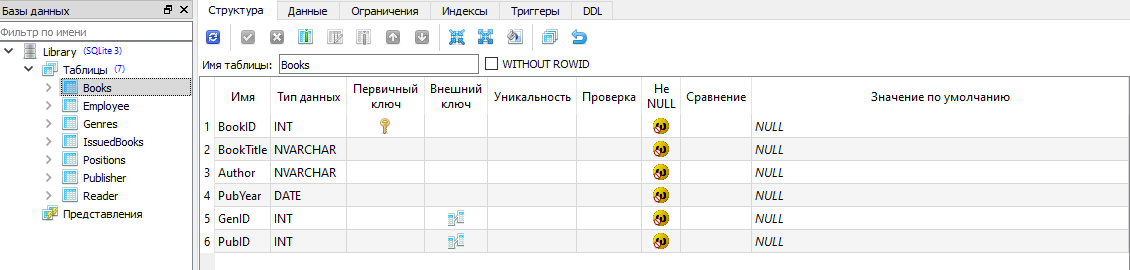


Рис. 1.39. Изменения типа данных столбца таблицы.

После всех преобразований в итоге будет файл базы данных, все типы столбцов которой поддерживаются в Visual Studio. На этом работа с SQLiteStudio заканчивается.

# **2 Создание приложения**

## **2.1 Создание проекта “Веб-приложение ASP.NET Core”**

Далее в работе будет использоваться Visual Studio 2019.

Перед началом работы нужно также убедиться, что все рабочие нагрузки установлены.

Для этого в Visual Studio Installer нужно нажать на “Изменить”.

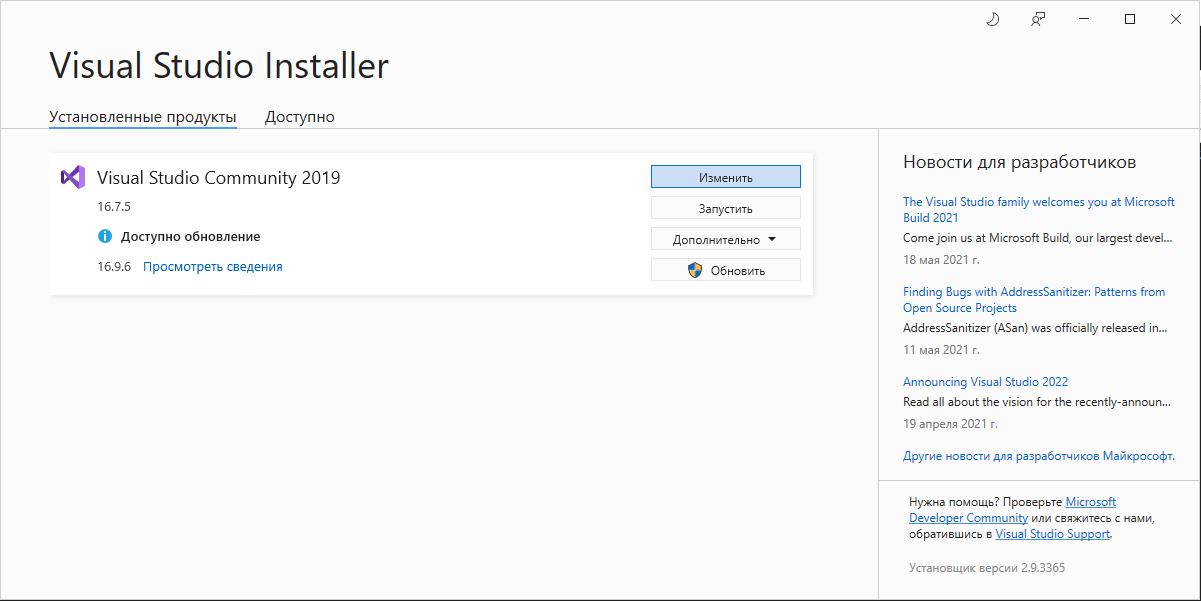


Рис. 2.1. Изменение доступной версии.

И выбрать следующие пункты.

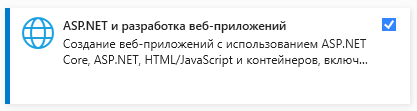


Рис. 2.2. Пакет «ASP.NET и разработка веб-приложений».

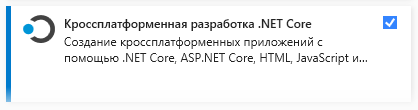


Рис. 2.3. Пакет «Кроссплатформенная разработка .NET Core».

При открытии Visual Studio нужно создать новое решение и выбрать “Веб-приложение ASP.NET Core”.

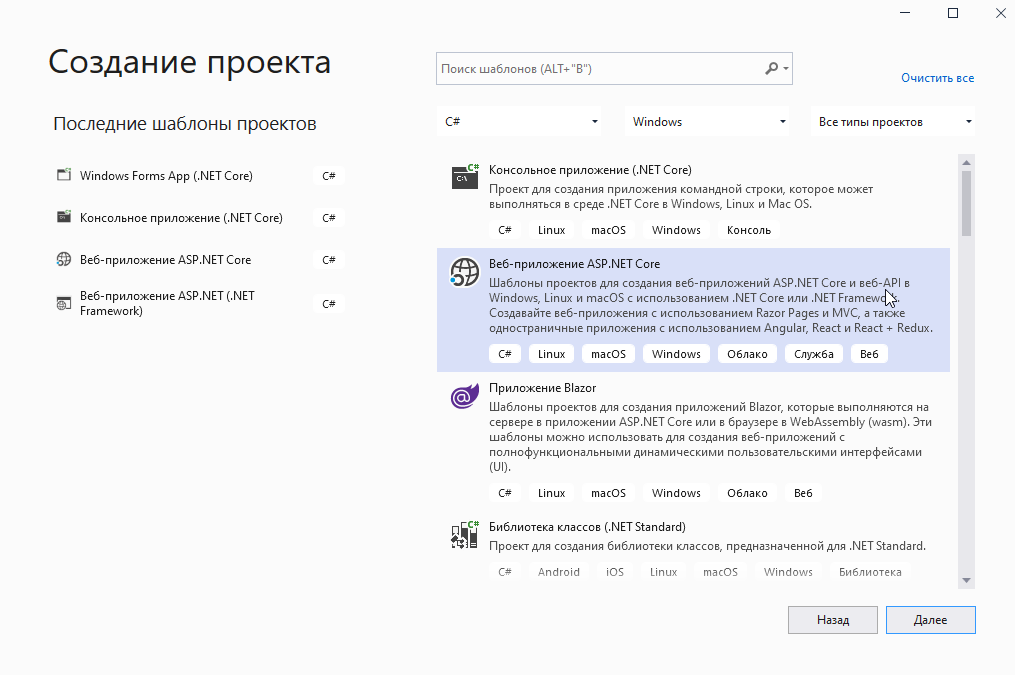


Рис. 2.4. Создание нового проекта, выбор шаблона.

Задаем имя и расположение проекта.

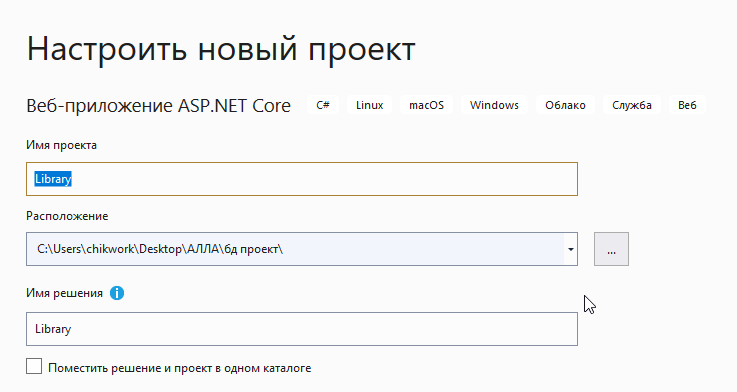


Рис. 2.5. Выбор имени решения и проекта, а также расположение.

В появившемся окне выбираем “Веб-приложение”.

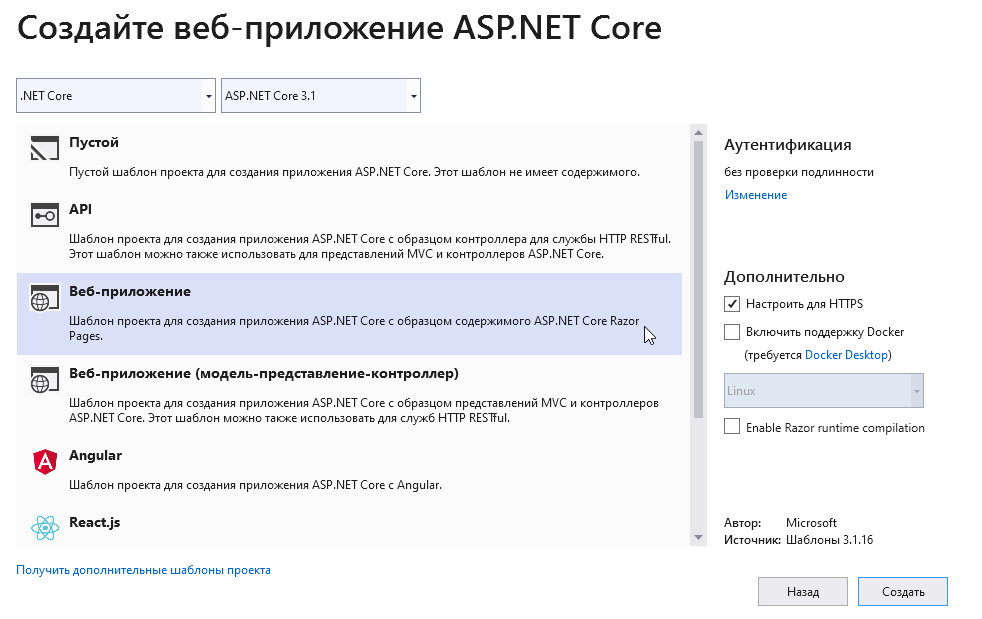


Рис. 2.8. Дополнительный выбор шаблона при выборе «Веб-приложение ASP.NET Core».

Cвяжем решение с GitHub. Для этого нажав на пункт “Файл” выберем пункт “Добавить в систему управления версиями”.

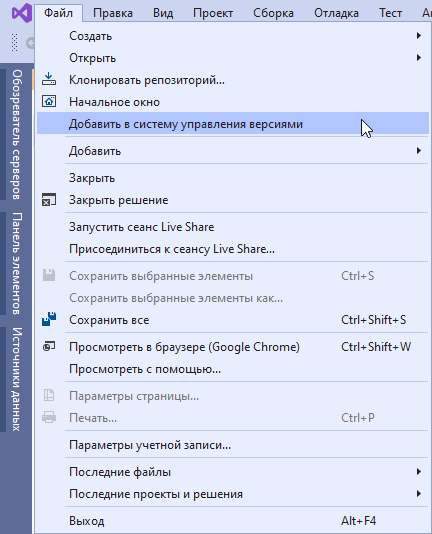


Рис. 2.9. Возможные действия над файлом.

Снизу приложения появится следующие кнопки:



Рис. 2.10. Возможные действия над файлом.

Нажав на стрелку, появится окно, в котором можно связать наш репозиторий с аккаунтом GitHub. Необходимо ввести URl адрес предварительно созданного на GitHub пустого репозитория.

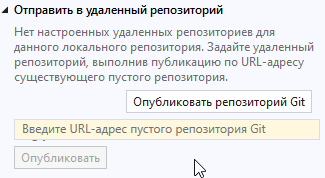


Рис. 2.11. Связь локального репозитория с аккаунтом GitHub.

Для работы нужно добавить все необходимые пакеты NuGet для проекта. Для этого, находясь во вкладке «Средства» необходимо выбрать «Управление пакетами NuGet для решения…»

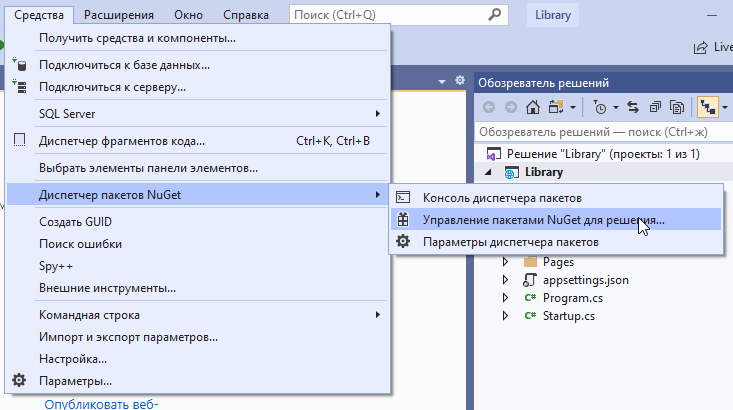
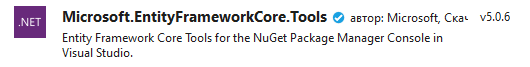
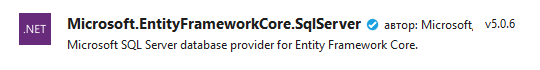
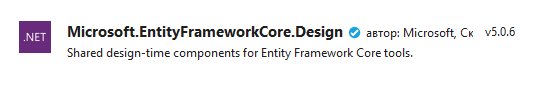
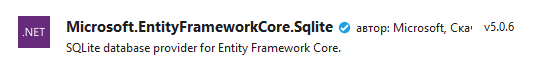


Рис. 2.11. Действия над решением.

После чего найти и добавить следующие пакеты:

## **2.2 Генерация контекста и моделей**

Для работы с базой данных в Visual Studio понадобятся модели и контекст. Их можно создать с помощью, созданного ранее в SQLiteStudio файла базы данных.

В “Консоли диспетчера пакетов” прописываю команду:

*Scaffold-DbContext ‘Data Source = \*путь к файлу\*\\*название файла\*.db’ Microsoft.EntityFrameworkCore.Sqlite -f -ContextDir Data -OutputDir Models*

Где *'Data Source=\*путь к файлу\*\\*название файла\*.db’* — источник данных, т.е. указываем путь к нашему файлу .db, а *-ContextDir Data -OutputDir Models* — это выходные директории для контекста и моделей.

При успешном исполнении этой команды у вас будет создана папка в решение под названием Models, в которой будут файлы с сущностными классами, построенными на основе таблицы базы данных, и класс контекста базы данных.

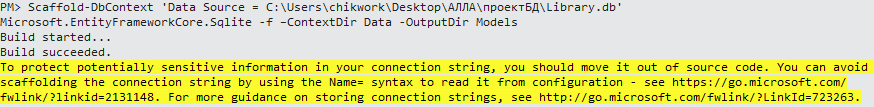


Рис. 2.12. Создание моделей и контекста с использованием ранее созданного файла базы данных.

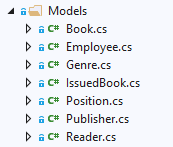


Рис. 2.13. Созданные модели.

В моделях меняем тип данных byte[] на DateTime.

## **2.3 Миграция и переход на локальный сервер**

При создании моделей с помощью команды scaffold, в контексте базы данных будет строка, указывающая на источник данных для базы.

*optionsBuilder.UseSqlite("Data Source=C:\\Users\\1\\Desktop\\ИСП-21\\proectdb\\Library\\Library.db");*

Чтобы переключится на локальный сервер Visual Studio, нужно заменить эту строчку на следующую:

*optionsBuilder.UseSqlite("Data Source= CHIK\\SQLEXPRESS; Database = Library.db; Integrated Security=True");*

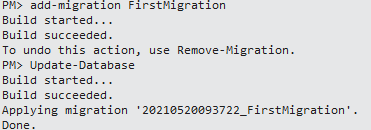
CHIK\\SQLEXPRESS – имя сервера конкретно в моем случае.

Также, в файле Startup.cs добавляем строчку:

*services.AddDbContext<InsuranceCompanyContext>(options =>  
options.UseSqlServer("Data Source=* CHIK\\SQLEXPRESS*;Initial Catalog=InsuranceCompany;Integrated Security=True"));*

Для сохранения изменений используется миграция.

В “Консоли диспетчера пакетов” пишем *Add-Migration \*имя миграции\**, а после обновляем базу данных с помощью команды Update-Database



2.14 успешная миграция

После этого в проекте будет создана папка Migrations.

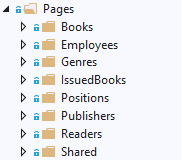


Рис. 2.15. Папка с миграциями в решении.

После успешного обновления база данных будет использовать самую последнею версию контекста.

## **2.4. Создание веб-страниц с представлениями и действиями**

Для создания веб-страницы, на которой будет отображаться таблица определённой сущности, нужно в готовой папке “Pages” создать папку для каждой сущности.



2.16 добавление папок

В таких папках нужно добавить страницу Razor.

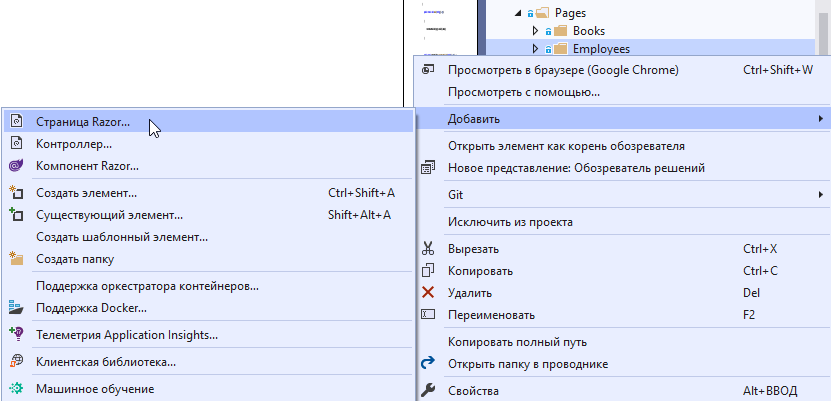


Рис. 2.17. Создание страницы Razor.

В качестве шаблона выбираем “Razor Pages на основе Entity Framework (CRUD)”.

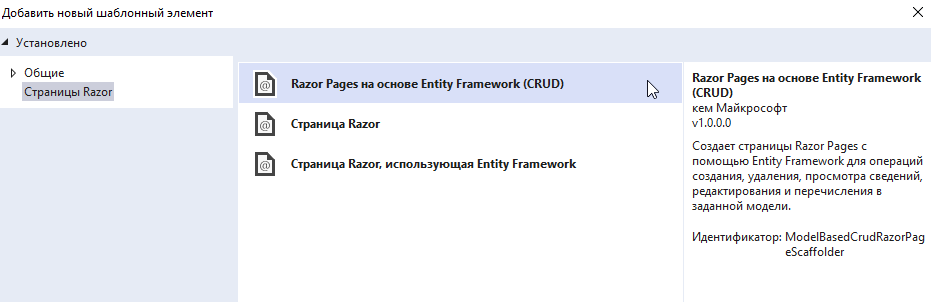


Рис. 2.18. Выбор шаблона страницы Razor.

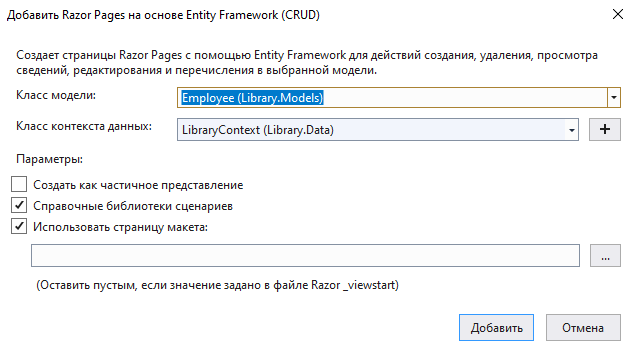
После чего необходимо указать соответствующий класс модели и контекста данных  


Рис. 2.19. Выбор модели и контекста для страницы Razor.

В папке будут созданы следующие файлы:

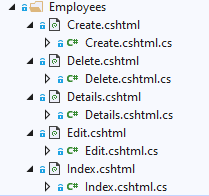


Рис. 2.20. Общий вид Razor страниц.

Файлы cshtml – это файлы представления, в них указывается как выводить и какие данные.

Файлы cshtml.cs – это файлы с запросами, они используются для получения данных для отображения.

При запуске приложения откроется главная страница.

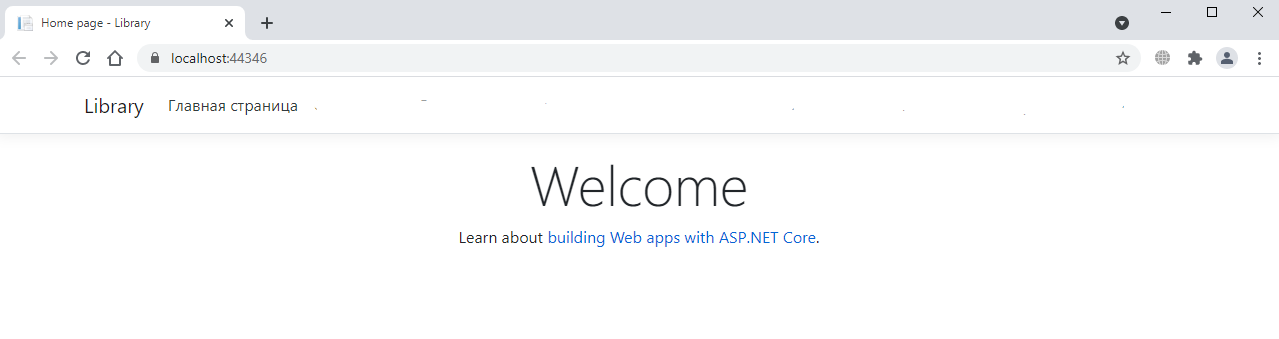


Рис. 2.21. Главная страница веб-приложения.

Чтобы в веб-приложении была ссылка на нашу веб-страницу, нужно зайти в папку “Pages”, в ней зайти в файл “Index.cshtml”. В нём прописываем код на добавление кнопок, с ссылками на папки со страницами сущностей.



Рис. 2.22. Создание ссылки на веб-страницу.

Для удобства восприятия, чтобы текст на сайте отображался на русском необходимо заменить выводимый текст в файлах .chtml. Чтобы таблицы также отображались на русском в моделях нужно прописать строчки [Display (Name = “выводимое имя”)]

Для использования этой функции необходимо использовать библиотеку System.ComponentModel.DataAnnotations.

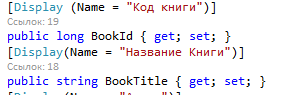


Рис. 2.23. Выводимое название столбца.

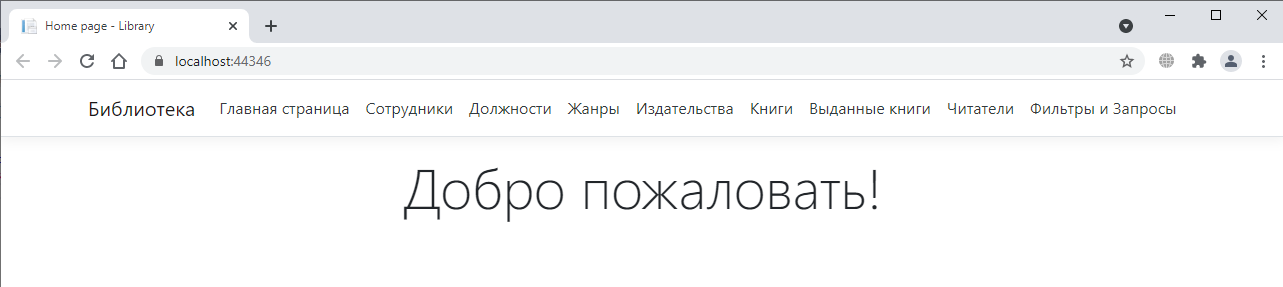


Рис. 2.24. Русифицированная главная страница приложения с добавленными кнопками

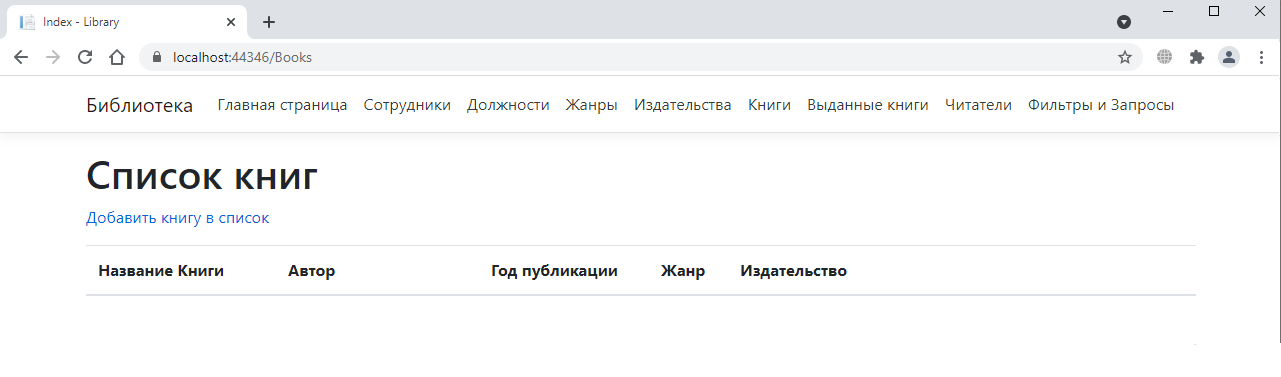


Рис. 2.25. Русифицированная веб страница со списком книг

После замены всего английского текста в cshtml файлах на русский и измены вида вывода переменных в файле сущности наша веб-станица будет завершена, так как весь функционал по добавлению, удалению и изменению таблицы был создан автоматически.

## **2.5 создание запросов и фильтров**

Для создания запросов и фильтров необходимо создать отдельную папку в папке страниц, в эту папку нужно добавить страницу Razor.

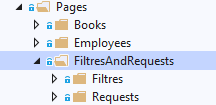


Рис. 2.26 добавление папок для фильтров и запросов

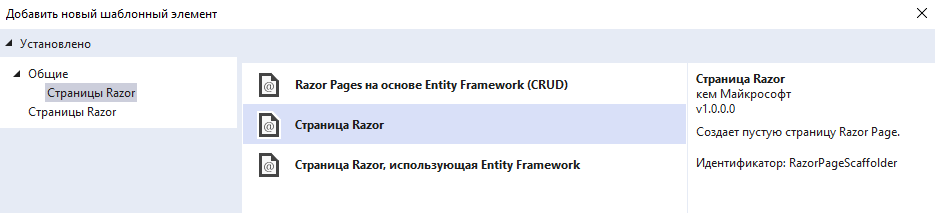


Рис. 2.27 добавление страницы Razor, выбор шаблона

Для создания конкретного запроса также необходимо создать страницу Razor в папке с запросами.

В cshtml.cs файле следует добавить контекст. Он создавался автоматически при создании прошлых страниц, так что код можно скопировать и подправить, чтобы он работал с моделью данного файла, а не другого.



Рис. 2.28 Добавление контекста для страницы запроса.

После этого нужно создать список сущностей, которые будут использоваться для вывода.

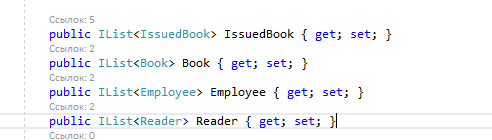


Рис. 2.29. Объявление сущностей для вывода.

После необходимо заменить автоматическую функцию “OnGet()” на “Task OnGetAsync()” и внутри функции сделать запрос на получения существующих данных об нужных сущностях.

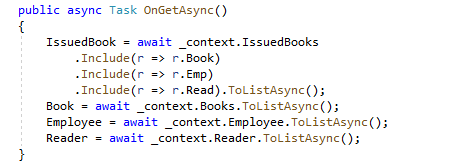


Рис. 2.30. Заполнение сущностей для вывода.

В Файле cshtml нужно написать код, который будет выводить на веб-странице нужные данные. Это можно сделать на основе кодов других файлов .cshtml.

Чтобы пользователь мог удобно воспользоваться запросами, необходимо также добавить кнопку с ссылкой на страницу с перечнем фильтров и запросов. Для этого в файле Layout.cshtml нужно добавить код на добавление кнопки

  
Рис. 2.31 добавление ссылок

Для отображения нашего запроса на данной странице, в папке FiltresAndRequests в файле Index.cshtml необходимо создать ссылки на наши запросы.



Рис. 2.32. Создание ссылок на страницы запросов

Для фильтров, также, как и для запросов, следует в отдельной папке создать страницу Razor.

Так как фильтры создаются на основе уже сделанных запросов, то для начала можно скопировать всё содержимое cshtml и cshtml.cs файлов запроса в файлы фильтра.

Вид, который можно использовать для фильтрации, может быть разный. В данном примере будет использоваться такой:

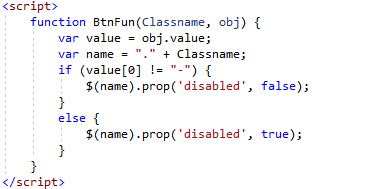


Рис. 2.33. Скрипт для фильтрации в файле страницы для фильтров и запросов

Он получает данные от списка и если первый символ у полученного значения будет равен “-“, то кнопка с переданным именем будет отключена.

В файле Index.cshtml добавляем ссылки на фильтры. Также создаётся текст, объясняющий на какой фильтр данная ссылка, список, в котором указаны все существующие ID указанной сущности, и кнопка.



Рис. 2.34. Создание ссылок на страницы фильтров.

Чтобы можно получить все существующие ID типа сущности, нужно в Index.cshtml.cs файле сделать запрос.

Создаётся список, в котором будут хранится ID типа сущности и в функции OnGet() происходит заполнение этого списка.



Рис. 2.35. Код списка для выбора должности, по которой будет проводиться фильтрование.

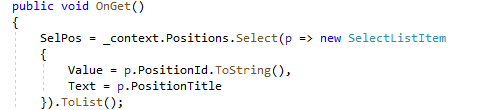


Рис. 2.36. Заполнение списка существующими “ID” должностей.

В переменной Value будет хранится значение ID, в переменной Text будет храниться текст, который будет выводиться

В итоге на странице с запросами и фильтрами будет создана форма, которая будет позволять показывать страницы с запросами только с указанным значением переменной.

Так как фильтры будут работать, используя один конкретный экземпляр сущности, то в файле cshtml.cs нужно заменить список этой сущности на единый экземпляр.

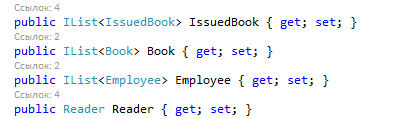


Рис. 2.37. Создание экземпляра сущности, по которому будет проводиться фильтрование.

Чтобы функция OnGetAsync могла возвращать страницы с разными значениями в таблицах, объявление функции нужно заменить на следующие.



Рис. 2.38. Изменённое объявление функции OnGetAsync.

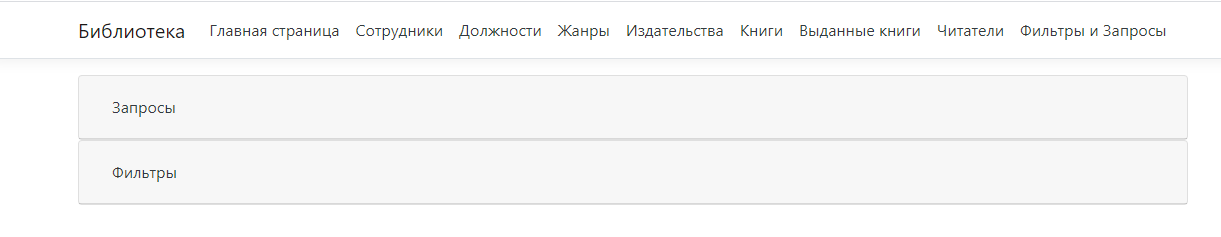


Рис. 2.39. Готовая страница с запросами и фильтрами.

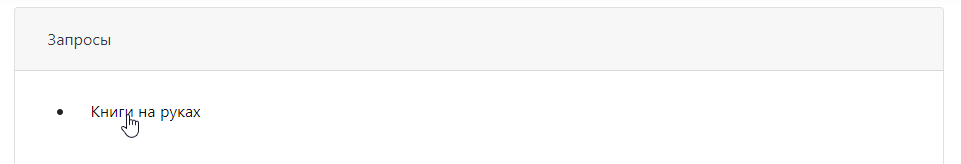
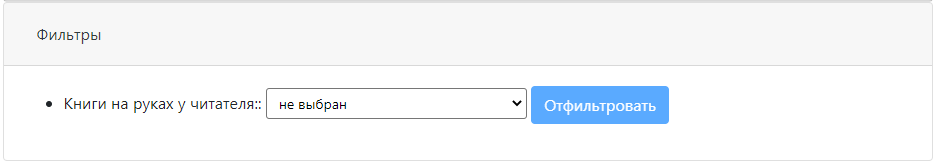


Рис. 2.40. запросы.

 Рис. 2.41. фильтры

# **Заключение**

По завершению работы был создано web-приложение, предназначенное для работы с базой данных библиотеки.

В процессе работы были рассмотрены:

* принципы работы с ER диаграммами и реляционными схемами
* создание и редактирование файла базы данных с помощью SQLiterStudio
* разработка web сайта в рабочей среде Visual Studio 2019 на основе модуля Entity Framework, с использование готового файла базы данных
* разработка и оформление страниц приложения, связь их с помощью ссылок
* создание фильтров и запросов

Приложение полностью русифицировано, позволяет добавлять, редактировать записи таблиц, а также редактировать вывод информации с помощью фильтров и запросов.

Реализованы запрос «Книги на руках» и Фильтр для отображения книг на руках отдельных читателей.

Источники

1. Адам Фримен «Entity Framework Core 2 для ASP.NET Core MVC для профессионалов»;
2. Документация Microsoft, по работе с Entity Framework: <https://docs.microsoft.com/ru-ru/ef/>;
3. Веб-сайт ERDPlus: <https://erdplus.com/>;
4. Официальный веб-сайт Microsoft: <https://visualstudio.microsoft.com/ru/>;
5. <https://sqlitestudio.pl> – официальный сайт SQliteStudio
6. <https://visualstudio.microsoft.com/ru/> - официальном сайте Microsoft, скачивание Visual Studio 2019