

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 62 страницы.

Выполнила:

студентка 2 курса

группы ИСП-21-19

Кузьмина А. Д.

Преподаватель:

Гуров В. В.

Химки, 2021

В базе данных будут следующие таблицы:  
Сотрудники (Код сотрудника, ФИО, Возраст, Пол, Адрес, Телефон, Паспортные данные, Код должности)[10 записей].

1. Должности (Код должности, Наименование должности, Оклад, Обязанности, Требования)[5 записей].
2. Издательства (Код издательства, Наименование, Город, Адрес)[5 записей].
3. Жанры (Код жанра, Наименование, Описание) [5 записей].
4. Книги (Код книги, Наименование, Автор, Код издательства, Год издания, Код жанра) [10 записей].
5. Читатели (Код читателя, ФИО, Дата рождения, Пол, Адрес, Телефон, Паспортные данные) [10 записей].
6. 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 ").

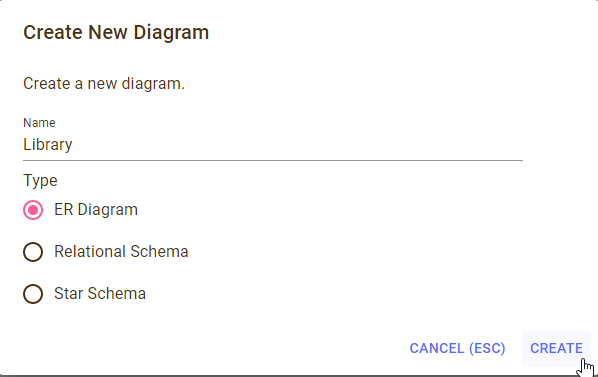
Создание базы данных

* 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>

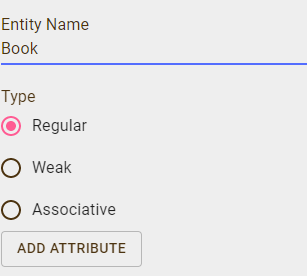
Чтобы начать работать с ER-диаграммами, нужно во вкладке “DOCUMENTS” нажать на кнопку “NEW DIAGRAM…”, выбрать “ER Diagram”, написать имя диаграммы и нажать на кнопку “CREATE”.

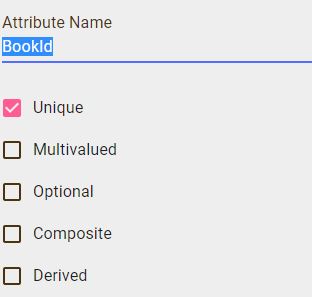


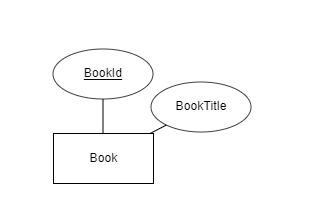


ER диаграмма состоит из сущностей, их атрибутов и связей. Сущности (Entity) в последствии будут преобразованы в таблицы базы данных, а атрибуты в поля этих таблиц.  
располагаются сущности в рабочей области с помощью кнопки Entity, после чего сущности задается необходимое имя.  


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



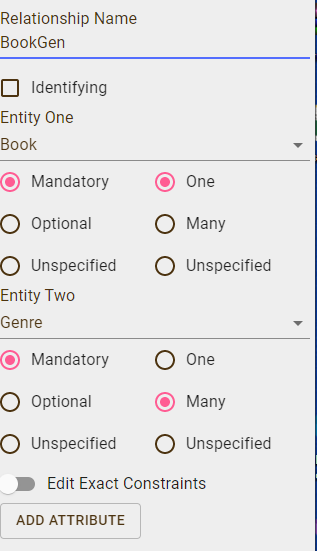
Для атрибутов-первичных ключей следует указать, что они уникальны.  


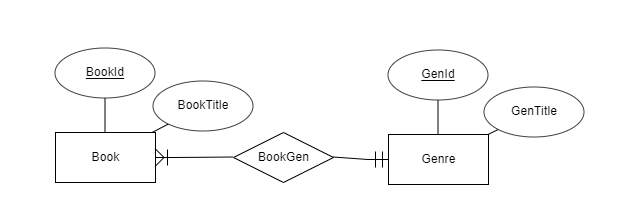


Чтобы связать сущности используется кнопка Connect.

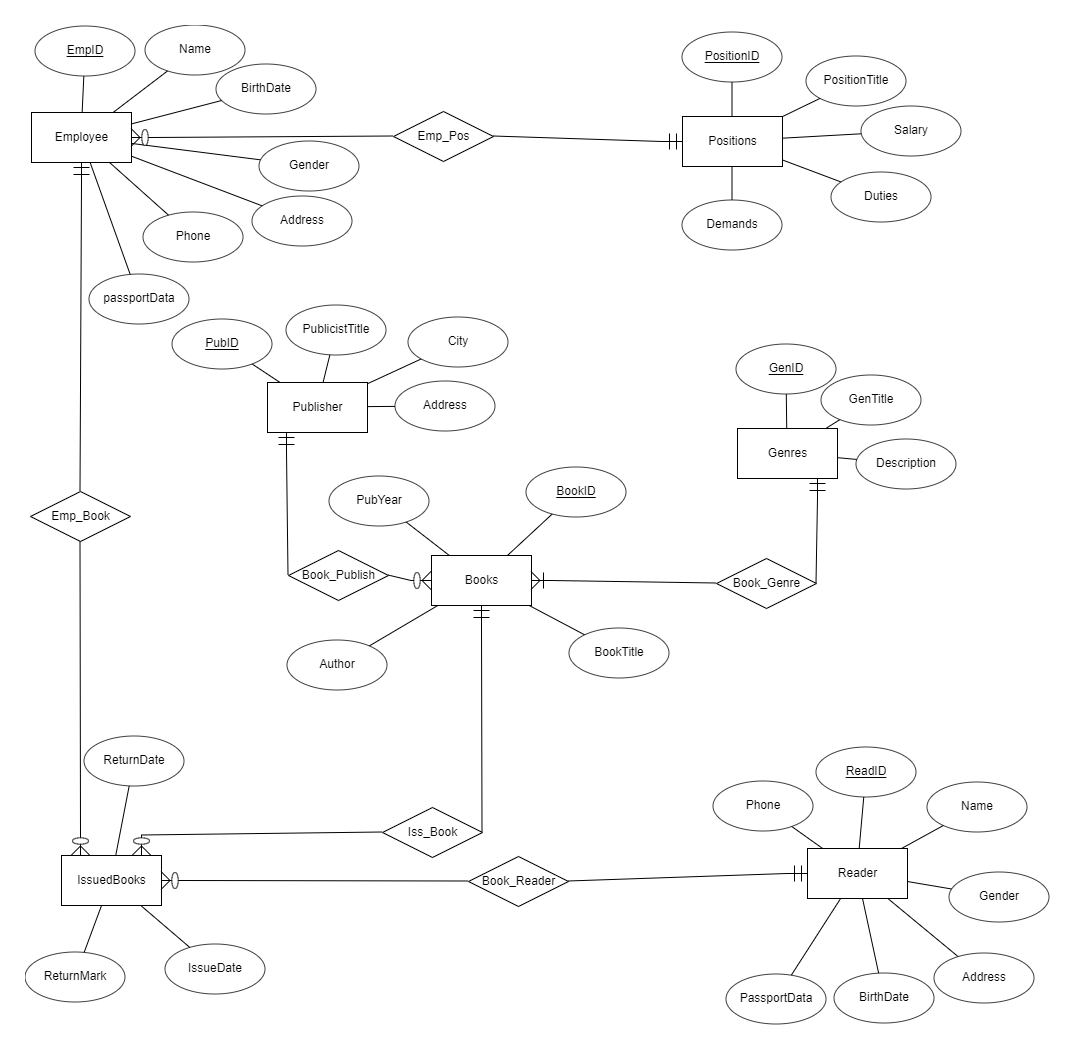


Для связи также можно задать имя. В настройках связи можно указать тип связи. Связь может быть “один к одному”, “один ко многим” и “многие ко многим”. Сущность в связи может быть *обязательной* или *необязательной.*





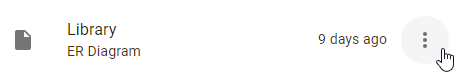
Готовая ER диаграмма для предметной области «библиотека»:

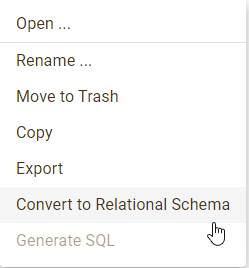


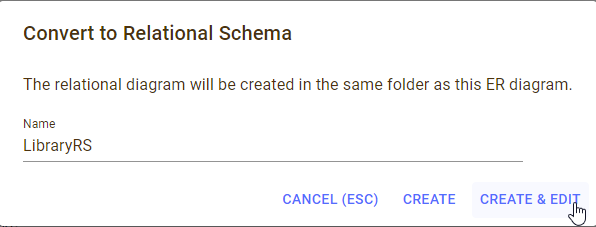
На этом работа с ER-диаграммой заканчивается и начинается работа с реляционной схемой.

* 1. Реляционная схема

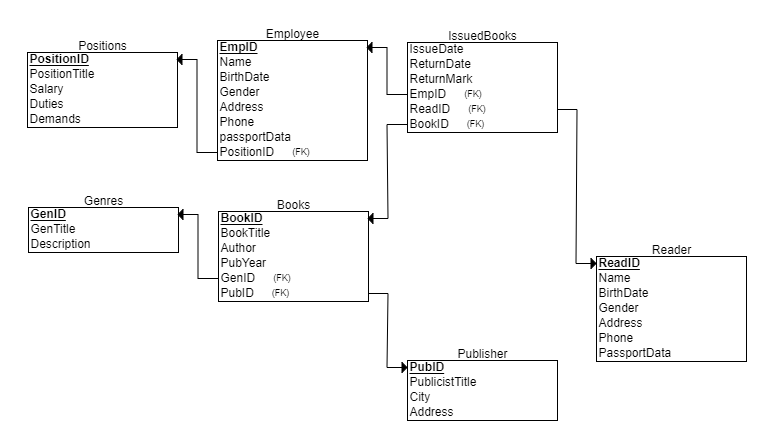
Для создания реляционной схемы используя существующую ER-диаграмму, находясь во вкладке “DOCUMENTS”, нужно найти необходимую диаграмму и в правой части экрана напротив её названия нажать на 3 точки, выбрать пункт “Convert to Relational Schema”, выбрать название схемы и нажать на кнопку “CREATE&EDIT”.





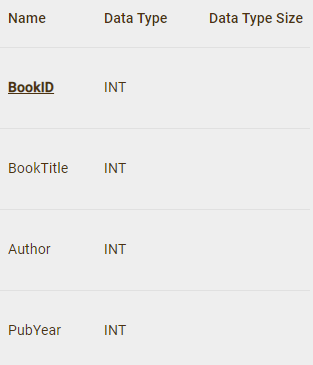


Сгенерированная реляционная схема:



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

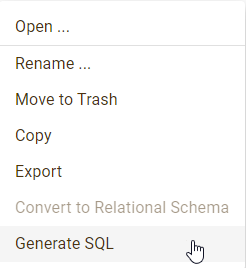
Также при работе с реляционной схемой у таблиц можно менять типы данных, нажав на сам тип напротив названия столбца.



На этом работа с реляционной схемой закончена.

* 1. 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”.

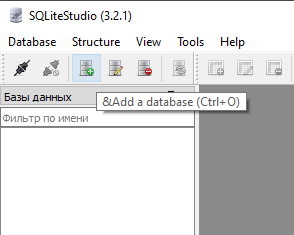


После чего необходимо создать файл с расширением .sql и вставить в него полученный sql скрипт.

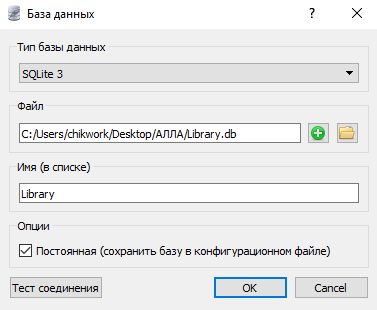
Для создания базы данных мы будем использовать приложение SQLite Studio.

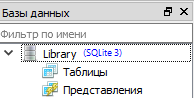
* 1. генерация базы данных

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

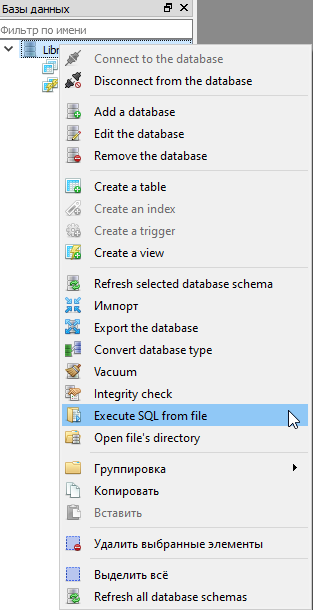


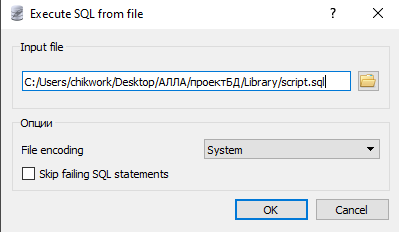
Далее необходимо задать имя файла и выбрать его расположение.

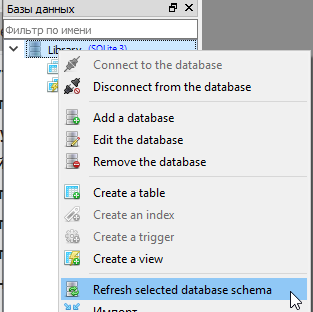




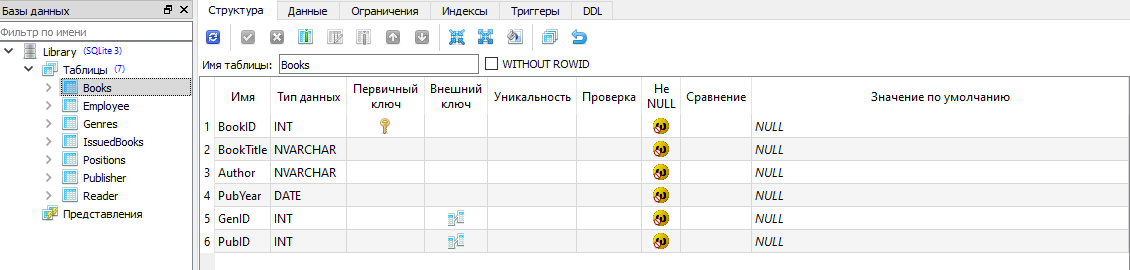
Чтобы воспользоваться SQL-скриптом для создания базы данных, нужно нажать правой кнопкой по базе и выбрать пункт “Execute SQL from file”.



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


После этого нужно обновить базу данных, нажав на кнопку “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.



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”.

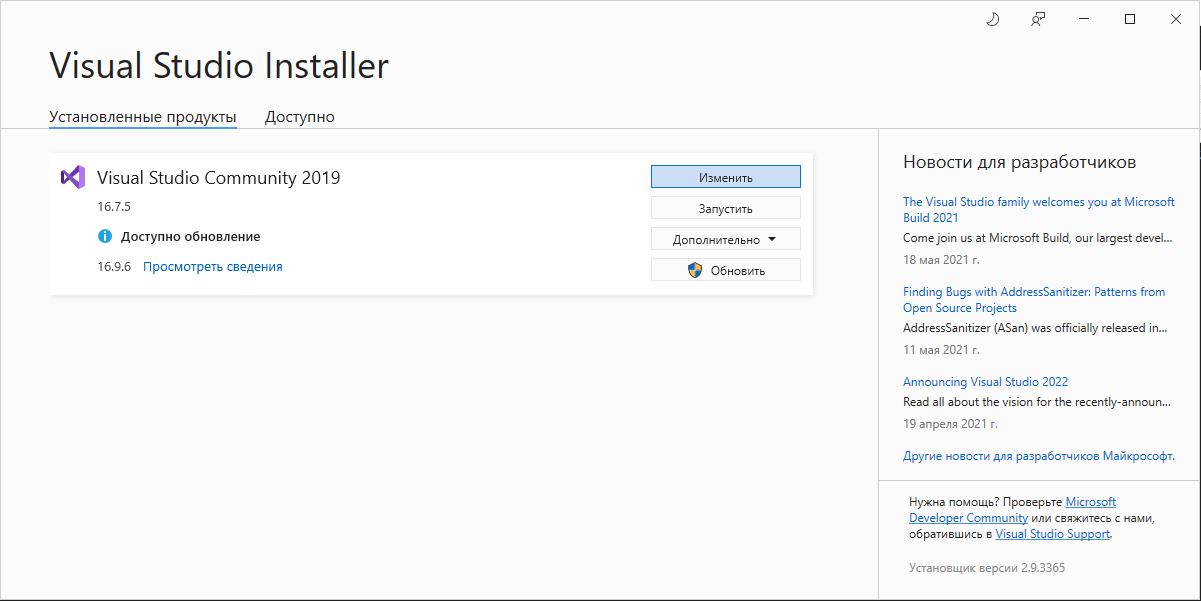


Рис. 2.2. Изменение доступной версии.

И выбрать следующие пункты.

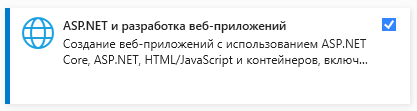


Рис. 2.3. Пакет «ASP.NET и разработка веб-приложений».

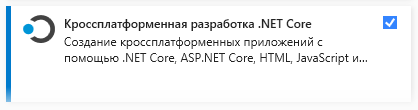
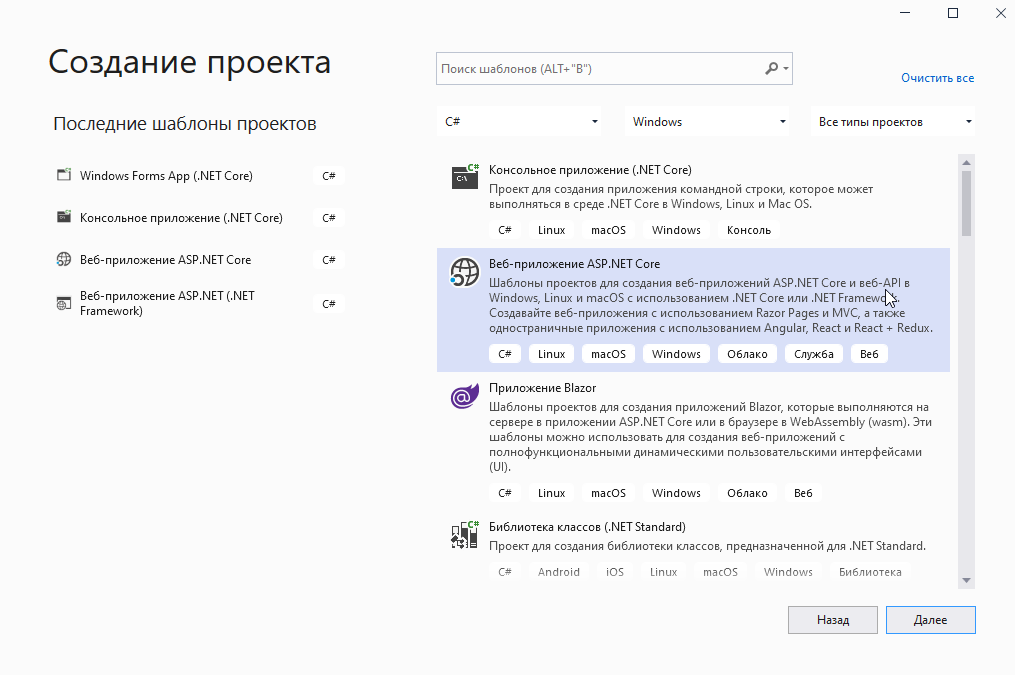
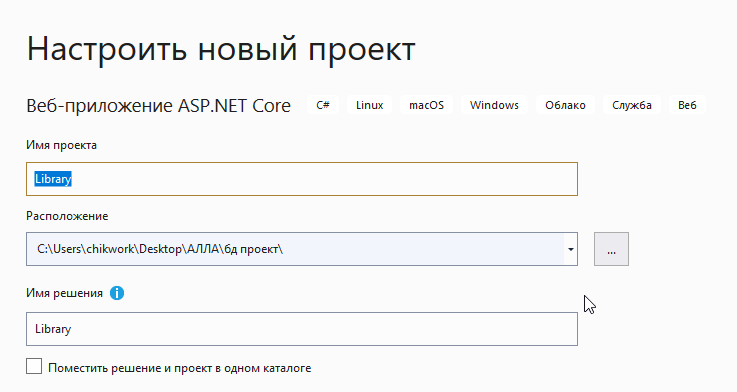


Рис. 2.4. Пакет «Кроссплатформенная разработка .NET Core».

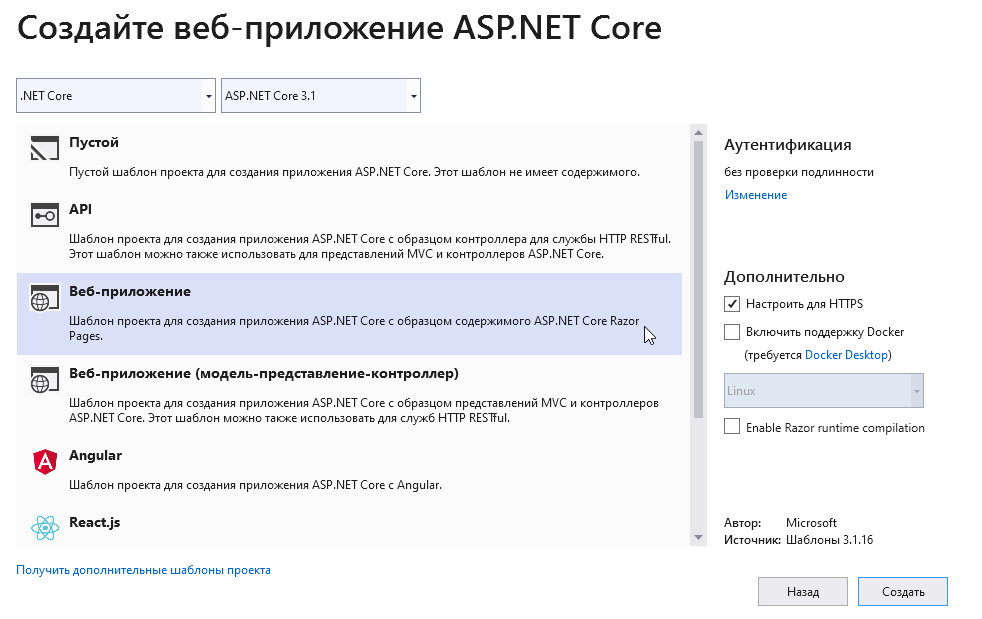
При открытии Visual Studio нужно создать новое решение и выбрать “Веб-приложение ASP.NET Core”.



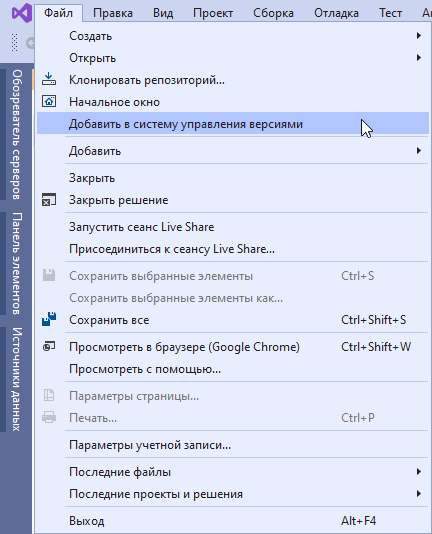
Задаем имя и расположение проекта.



В появившемся окне выбираем “Веб-приложение”.

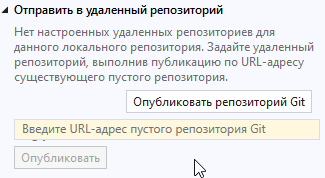


Cвяжем решение с GitHub. Для этого нажав на пункт “Файл” выберем пункт “Добавить в систему управления версиями”.

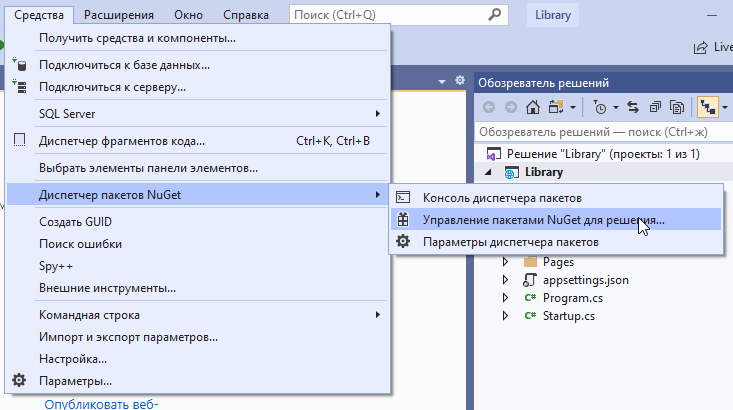


Снизу приложения появится следующие кнопки:

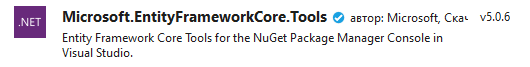
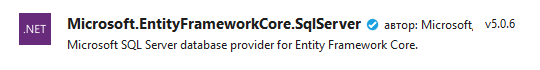
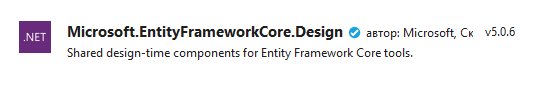
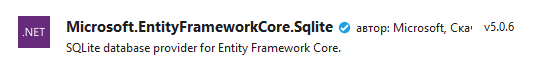


Нажав на стрелку, появится окно, в котором можно связать наш репозиторий с аккаунтом GitHub. Необходимо ввести URl адрес предварительно созданного на GitHub пустого репозитория.  


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



После чего найти и добавить следующие пакеты:



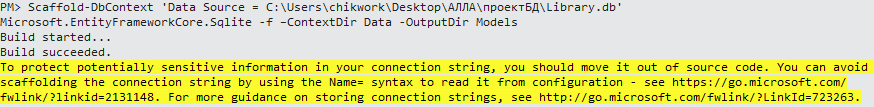
2.2 Генерация контекста и моделей

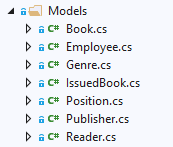
Для работы с базой данных в Visual Studio понадобятся модели и контекст. Их можно создать с помощью, созданного ранее в SQLiteStudio файла базы данных.

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.





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");

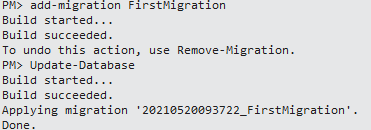
CHIK\\SQLEXPRESS – имя сервера конкретно в моем случае.

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



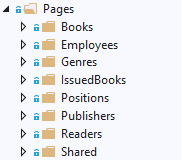
После этого в проекте будет создана папка Migrations.



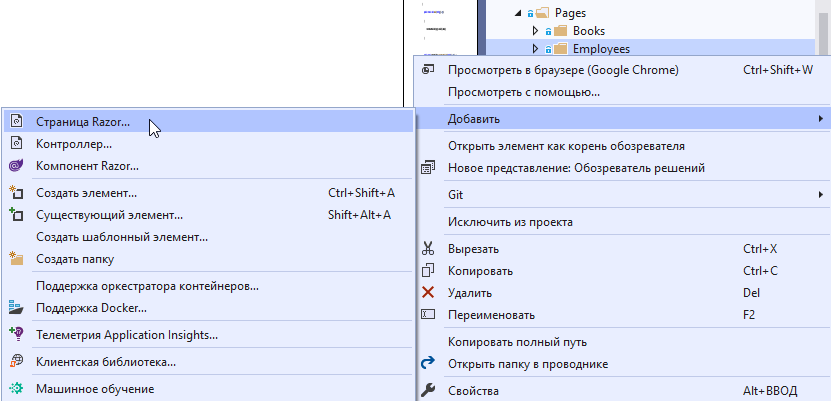
После успешного обновления база данных будет использовать самую последнею версию контекста.

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

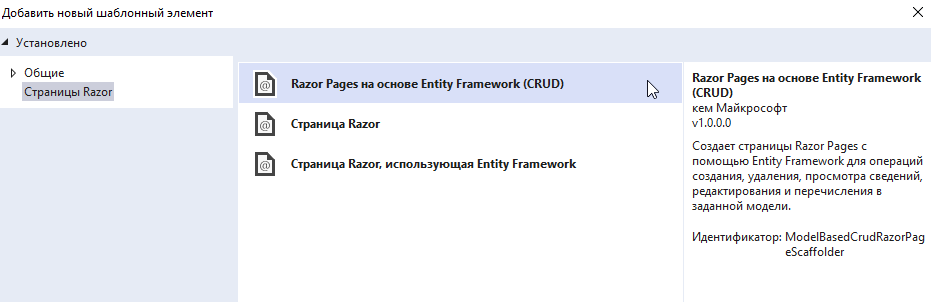
Для создания веб-страницы, на которой будет отображаться таблица определённой сущности, нужно в готовой папке “Pages” создать папку для каждой сущности.

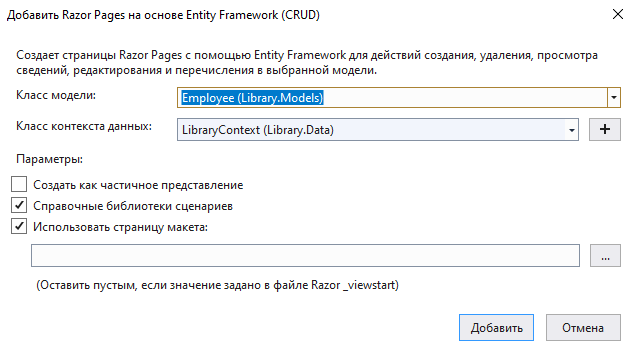


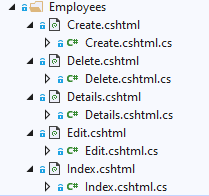
В таких папках нужно добавить страницу Razor.



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

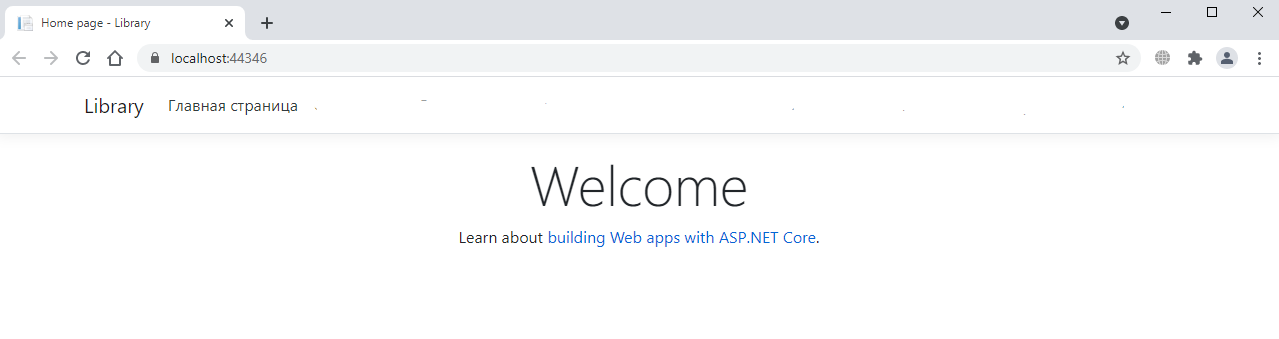


После чего необходимо указать соответствующий класс модели и контекста данных  


В папке будут созданы следующие файлы:  


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.

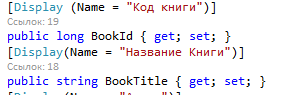
При запуске приложения откроется главная страница.

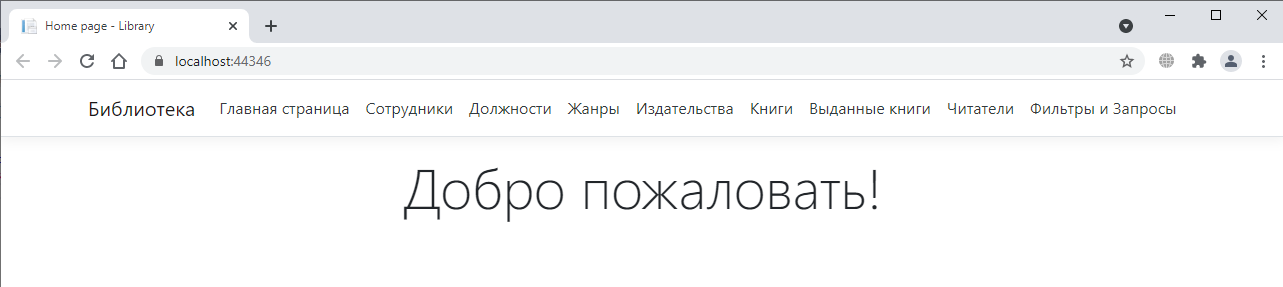
Чтобы в веб-приложении была ссылка на нашу веб-страницу, нужно зайти в папку “Pages”, в ней зайти в файл “Index.cshtml”. В нём прописываем код на добавление кнопок, с ссылками на папки со страницами сущностей.

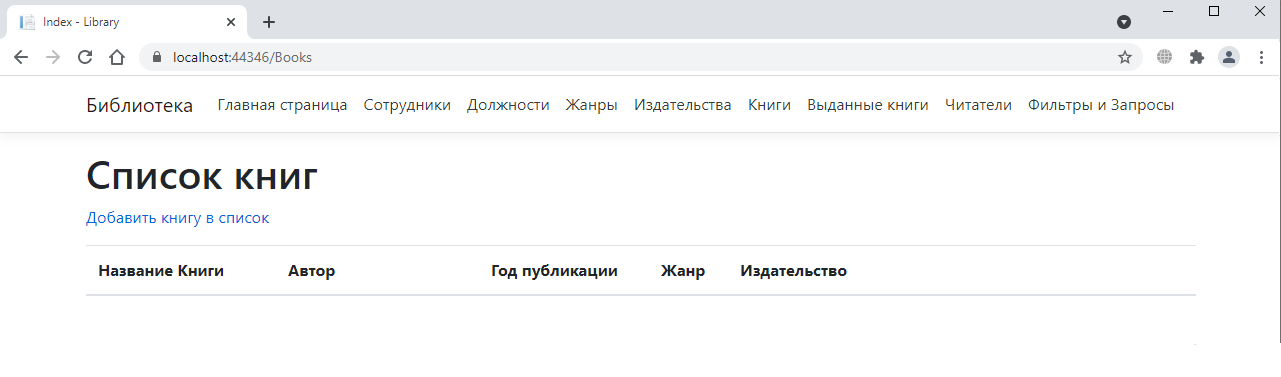


Для удобства восприятия, чтобы текст на сайте отображался на русском необходимо заменить выводимый текст в файлах .chtml. Чтобы таблицы также отображались на русском в моделях нужно прописать строчки [Display (Name = “выводимое имя”)]

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







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.

