# Разработка Веб-приложения

## Средства разрабтки

Для разработки веб-приложения выбраны следующие средства:

* Microsoft Visual Studio 2022 Community – бесплатная версия среды разработки Visual Studio для разрабтки приложений на платформа .Net Framework;
* SQLiteStudio – средство создания баз данных встраиваемой СУБД SQLite с графическим интерфейсом;
* Entity Framework – объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft.

## Создание нового проекта

Запустим Visual Studio и создадим новый проект веб-приложения. В качестве шаблона выбрано приложение ASP.NET Core MVC(модель-представление-контроллер) на платформе .NET 6.0 с долгосрочной поддержкой. В созданном шаблоне удалим автоматически сгенерированный контроллер HomeController и соответствующие ему представления.

## Подключение базы данных

Для работы с базой данных добавим в проект каталог App\_Data и скопируем в него файл базы данных, созданный с помощью SQLiteStudio. Далее добавим строку подключения в файл конфигурации appsettingы.json. Для работы с Entity Framework установим в приложение необходимый набор пакетов через менеджер пакетов NuGet:

* Microsoft.EntityFrameworkCore.Design;
* Microsoft.EntityFrameworkCore.Sqlite.

После установки пакетов для создания модели базы данных с помощью Entity Framework откроем утилиту Developer PowerShell через меню Visual Studio Средства->Командная строка. В командной строке выполним следующую команду «dotnet ef dbcontext scaffold --project WebApplication Name=sqlite Microsoft.EntityFrameworkCore.Sqlite --output-dir Models --context-dir Data --namespace WebApplication.Models --context-namespace WebApplication.Data --context SqliteContext -f --no-onconfiguring». После выполнения команды в проекте будут автоматически сгенерированы классы объектной модели базы данных с учетом отношений между таблицами и необходимые классы для подключения и выполнения команд с БД.

## Инициализация подключения

После создания объектной модели необходимо добавить код инициализации подключения, для этого откроем файл Program.cs и добавим следующую строку «builder.Services.AddDbContext<SqliteContext>(options=> options.UseSqlite("Name=sqlite"))».

## Создание контроллера Home

В обозревателе решений выберем папку Cotrollers и добавим в него пустой контроллер Home. Контроллер Home и действие Index вызываются по умолчанию при открытии пользователем веб-приложения, поэтому он будет использоваться в качестве контроллера авторизации пользователей.

Для доступа к БД из контроллера добавим поле «SqliteContext \_context» (SqliteContext – сгенерированный Entity Framework класс, реализующий функции чтения и записи данных в БД), отметим поле модификаторами «private» и «readonly», и добавим конструктор с таким же параметром и запишем параметр в поле.

Добавим для действия Index представление Index.cshtml с формой авторизации. Для обработки формы добавим в контроллер действие Index с параметрами, передаваемыми из формы, и отметим его как «HttpPost».

## Авторизация пользователей

Обработка формы авторизации проводит поиск пользователя по введенным логину и паролю среди всех пользователей. Поскольку пароль хранится в БД в зашифрованном виде, добавим в папку Models проекта статический класс Helper для реализации методов, необходимы в разных частях приложения во избежание дублирования функций.

В класс Helper добавим статический метод CreateMD5 для шифрования введенного на форме авторизации пароля по алгоритму MD5.

Для хранения данных авторизованного пользователя на время работы приложения запишем его данные, найденные в коллекции пользователей, в сессию.

Для работы с сессией добавим в файл Program.cs строку «builder.Services.AddSession(options => { options.IdleTimeout = TimeSpan.FromHours(1);});». Свойство IdleTimeout определяет время хранения сессии в памяти если пользователь не совершает никаких действий.

## Ограничение доступа неавторизованных пользователей

Доступ к функционалу веб-приложения может осуществляться не только через интерфейс html-страниц, но и через адресную строку браузера. Для того чтобы пользователь не мог обращаться к функциям приложения без авторизации через адресную строку создадим метод ShowPage в классы Helper и передадим ему текущую сессию приложения и имя запрашиваемой страницы. Метод будет возвращать значения true/false в зависимости от доступности функционала пользователю.

Данных метод будет вызываться во всех контроллерах перед загрузкой страниц или выполнением операций, и перенаправлять пользователя на страницу авторизации или иную доступную страницу.