Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  |  |

ИНСТРУКЦИЯ

по размещению API и БД в docker

|  |  |
| --- | --- |
|  |  |
|  | Исполнитель:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Южаков Д.С.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |  |

2023

# Необходимые условия

Нам понадобятся следующие вещи:

* Компьютер с Windows, Linux или Mac
* Редактор кода .NET (Visual Studio 2022)
* Движок Docker Engine (Docker Desktop)
* СУБД MySQL Workbench

# Создание проекта и подключение его к БД

Создаем новый проект веб-API в Visual Studio (Рисунок 1.1).

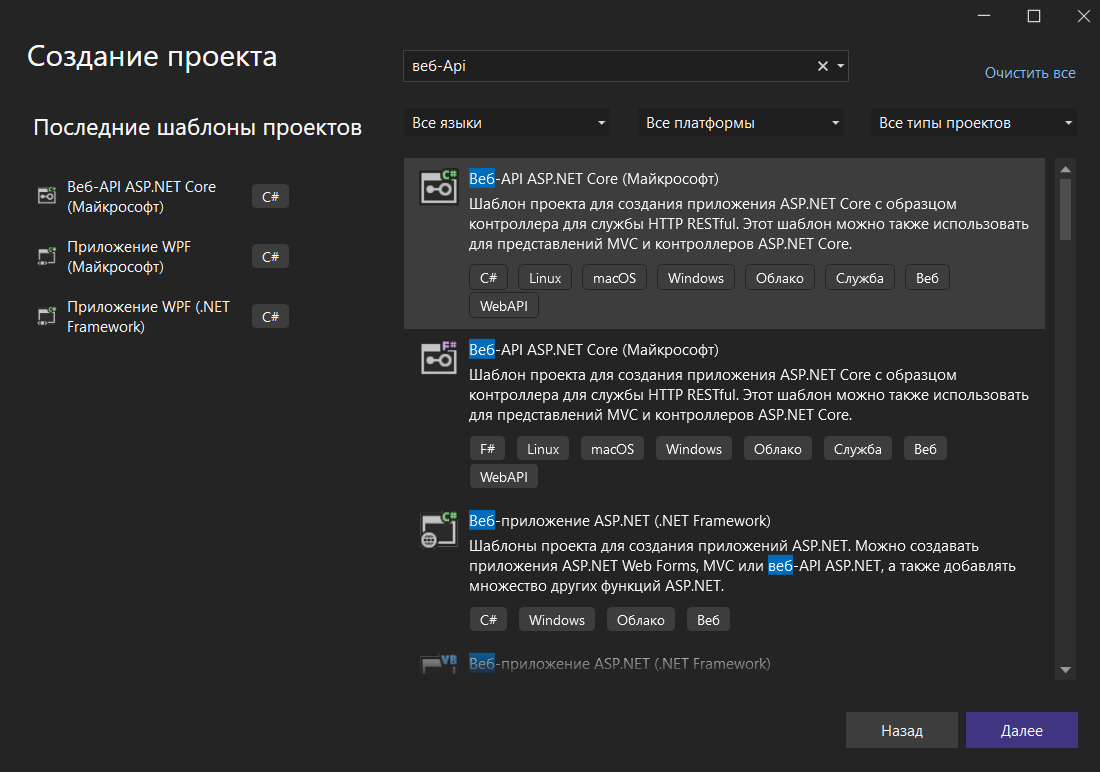


Рисунок 1.1 – Создание проекта

Добавить поддержку Docker можно при создании приложения, но это можно сделать и позже в самом проекте (Рисунок 1.2).

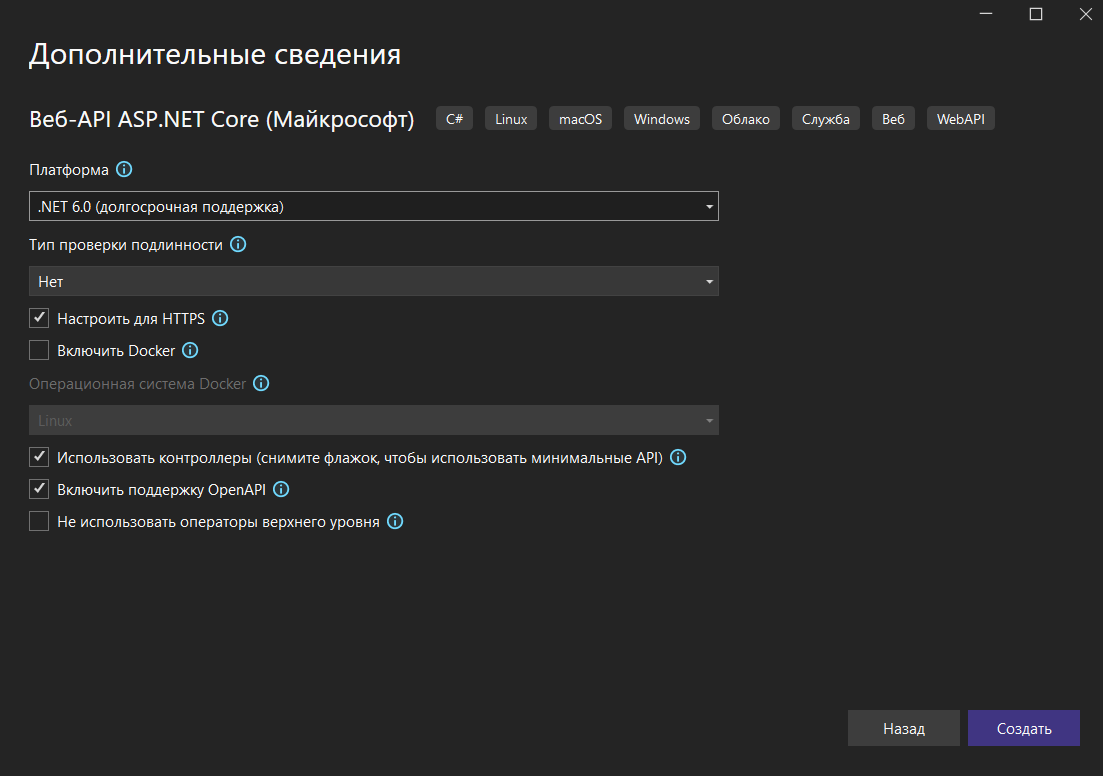


Рисунок 1.2 – Создание проекта

После создания проекта подключаем следующие NuGet пакеты:

* Pomelo.EntityFrameworkCore.MySql
* Microsoft.EntityFrameworkCore.Proxies
* Microsoft.EntityFrameworkCore.Tools

Далее с уже заранее готовую схему базы данных подключаем с помощью консоли диспетчера пакетов

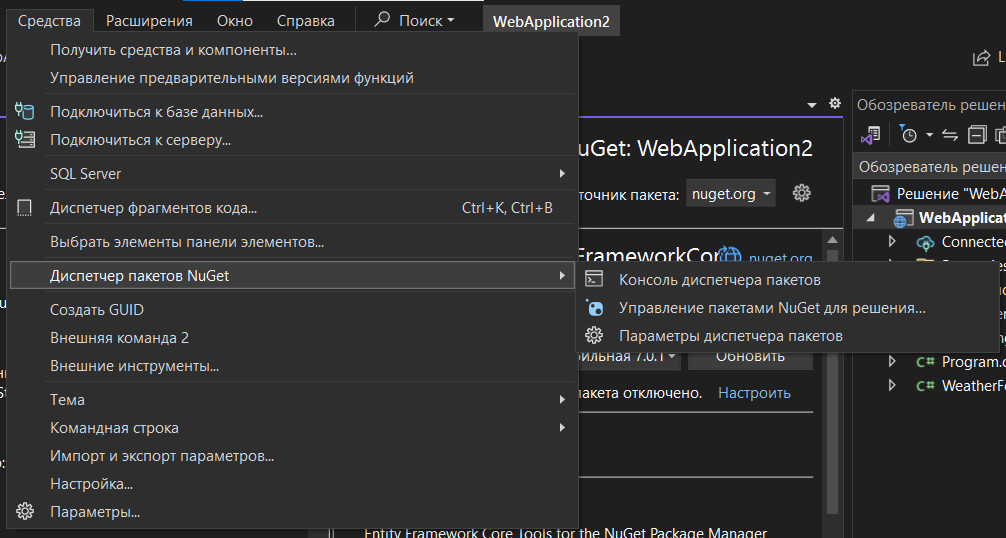


Рисунок 1.3 – Консоль диспетчера пакетов

В появившейся консоли нужно ввести следующую строку:

Scaffold-DbContext "server=localhost;user=root;password=1941;database=testschema" "Pomelo.EntityFrameworkCore.MySql" -OutPutDir "Model"

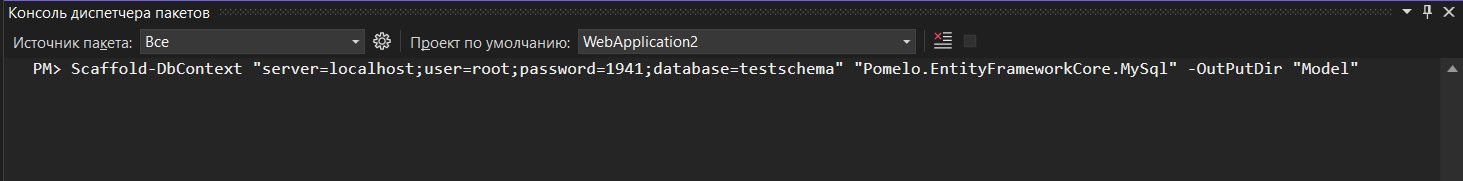


Рисунок 1.4 – Scaffolding

Добавим класс контекста базы для получения данных из нее и добавим сам констекст (Рисунки 1.5, 1.6).

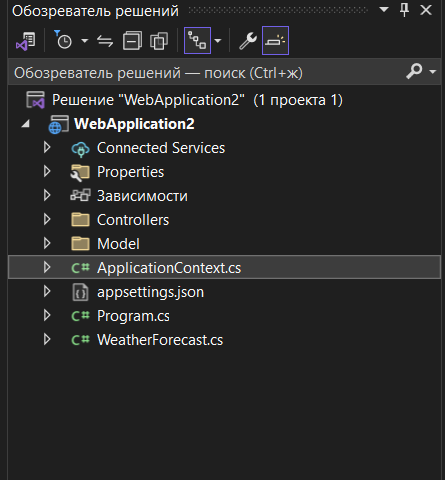


Рисунок 1.5 – Создание класса контекста БД

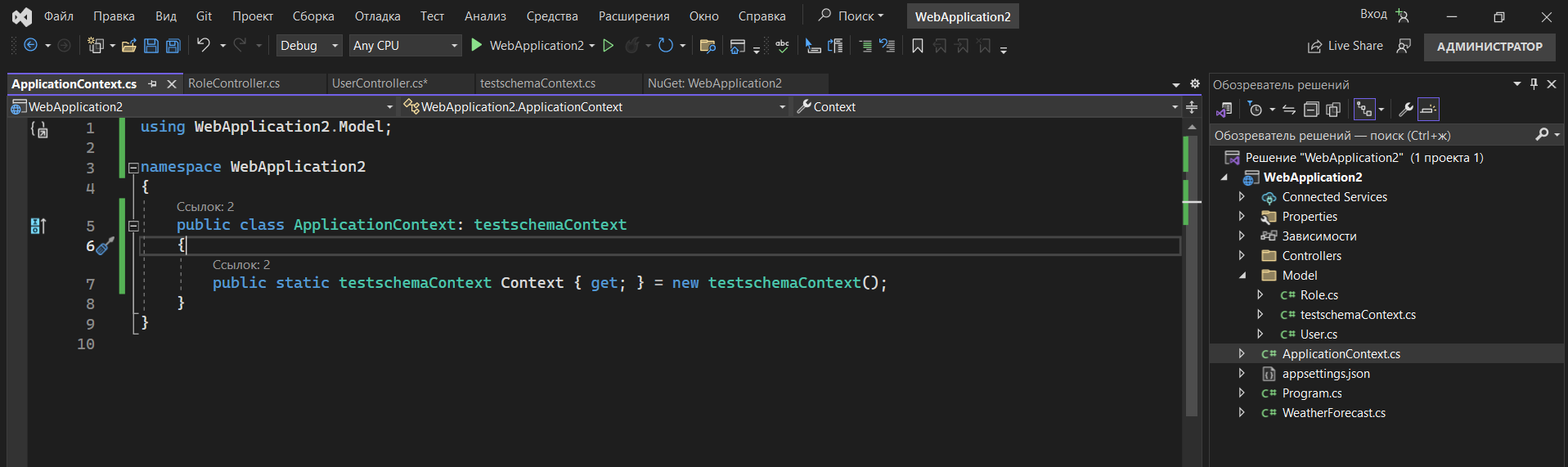


Рисунок 1.6 – Создание контекста

Создадим пустой API – контроллер и заполним его следующими методами изображенных на рисунке 1.7.

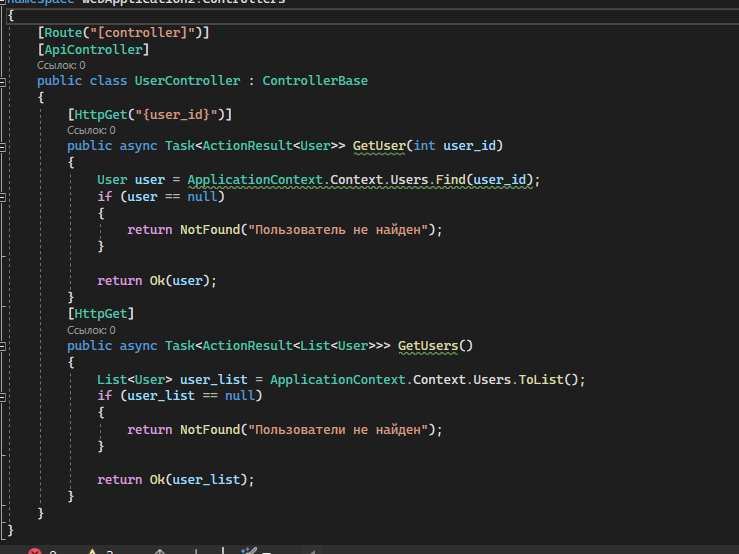


Рисунок 1.7 – Заполнение контроллера

Запустив проект, мы убеждаемся, что все работает и переходим к следующему шагу.

# Подключение Docker и Docker-Compose

Нажав ПКМ по проекту в обозревателе решений и, наведем на добавить мы увидим, что можем подключить поддержку Docker и Docker-compose (Рисунок 2.1). Подключим их и увидим что нас просят выбрать целевую OC. В нашем случае нужно выбрать Linux в обоих случаях (Рисунок 2.2).

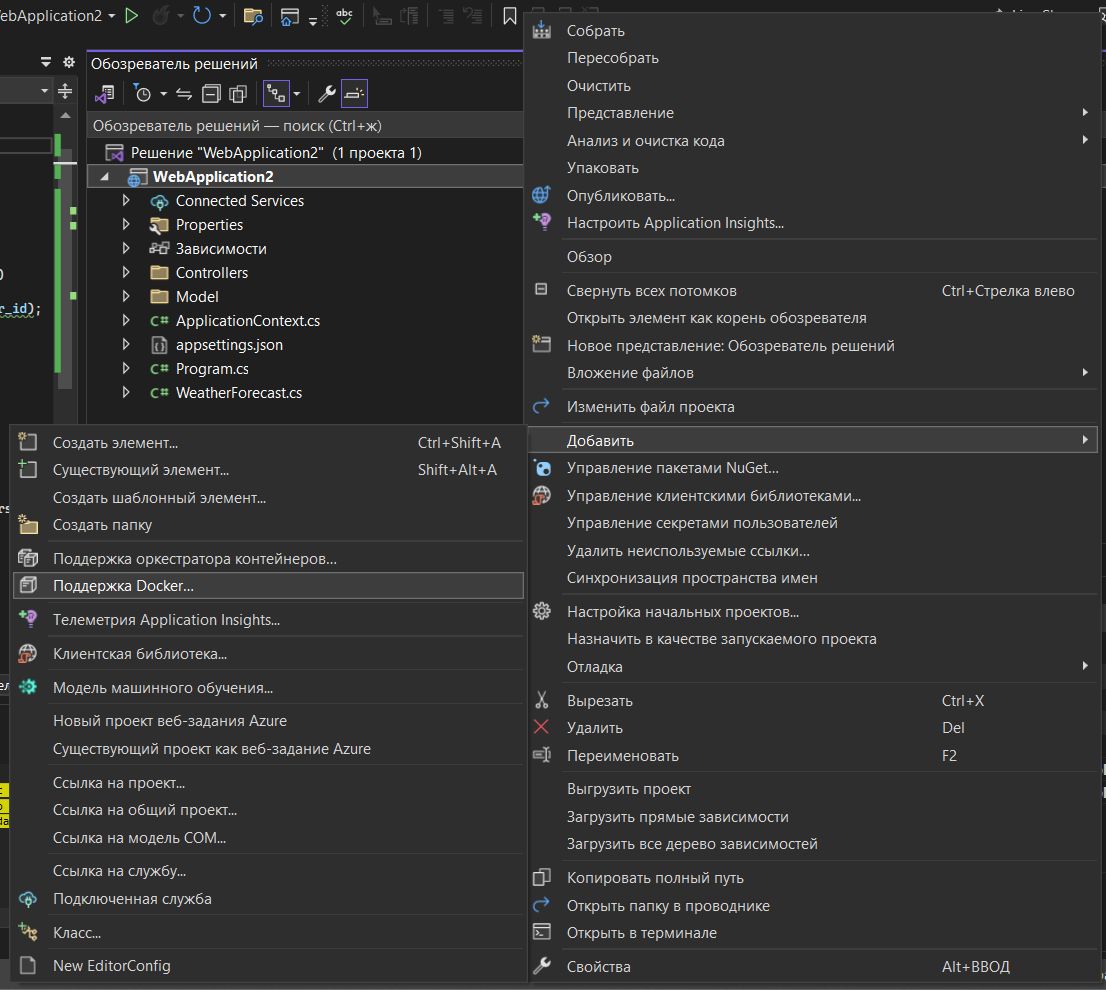


Рисунок 2.1 – Добавление поддержки docker и docker-compose

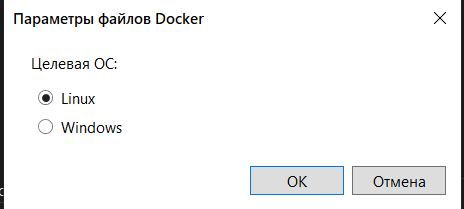


Рисунок 2.2 – Выбор целевой ОС

После их подключения мы видим, что в папке проекта появился файл с именем Dockerfile и второй проект с именем docker-compose (рисунки 2.3).

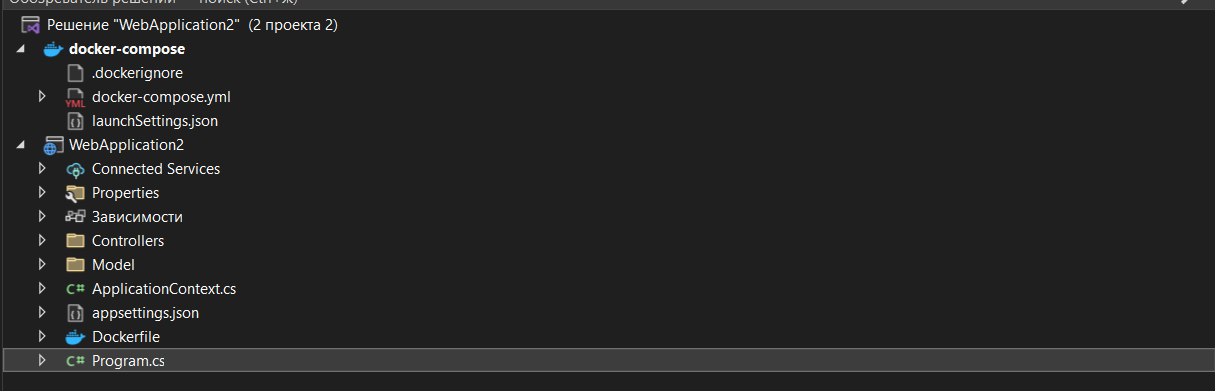


Рисунок 2.3 – Обозреватель решений после добавления поддержки docker

Откроем файл docker-compose.yml который находится в проекте docker-compose (Рисунок 2.4).

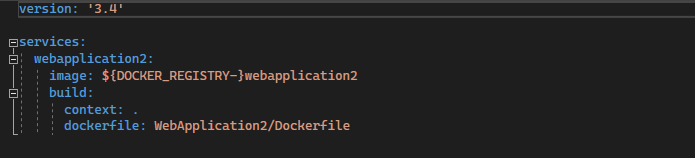


Рисунок 2.4 – Файл docker-compose.yml

В него нужно заполнить следующие данные, изображенные на рисунке 2.5.

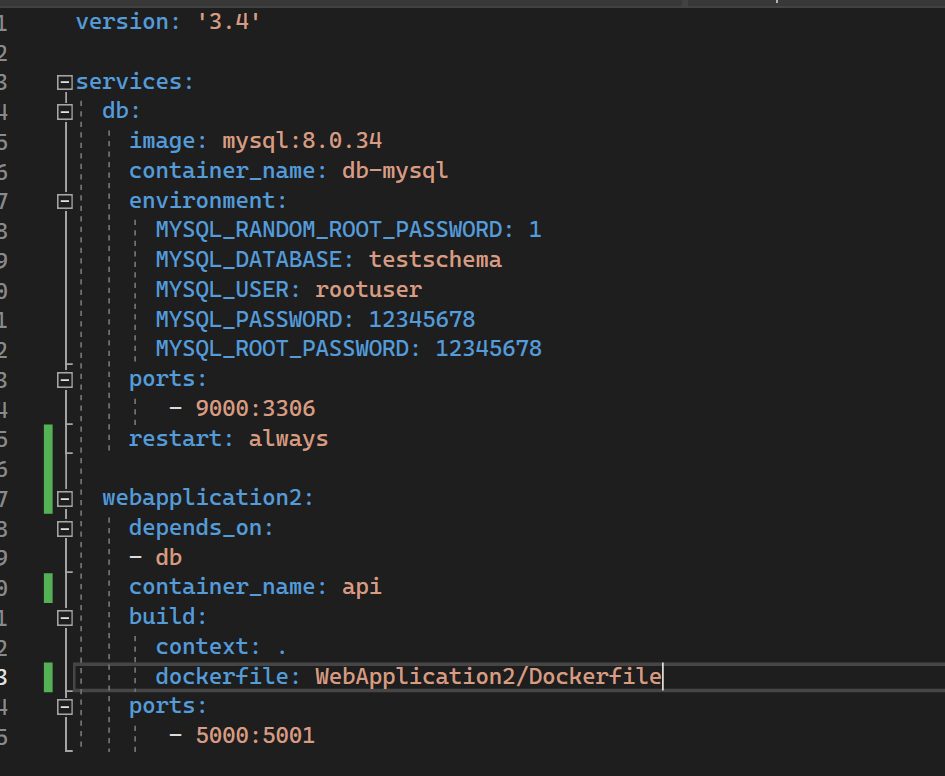


Рисунок 2.5 – Файл docker-compose.yml

Db – это сервис который мы добавили в docker-compose.yml. В службе базы данных мы настраиваем контейнер MySQL. Поле image указывает на образ СУБД и ее версию, если в поле указать образ без версии (image: mysql), то будет загружена последняя доступная версия в Docker Hub, мы же собираемся использовать образ mysql:8.0.34.

Поле container\_name указывает на имя контейнера, к которому мы будем обращаться т.к. оно является именем сервера для этого контейнера.

Образ MySQL имеет некоторые предопределенные значения среды, которые он ищет при инициализации.

Например:

* MYSQL\_RANDOM\_ROOT\_PASSWORD: 1 устанавливает случайный пароль для пользователя root.
* MYSQL\_DATABASE: testschema создает базу данных «testschema»
* MYSQL\_USER: root\_user создает пользователя «root\_user»
* MYSQL\_PASSWORD: 12345678 генерирует пароль для пользователя «root\_user»

Поле restart: always указывает Docker Compose перезапустить нашу службу в случае, если контейнер по какой-либо причине выйдет из строя.

Поля ports указывают внешний порт для подключения и локальный порт между службами. В этом случае локальный порт у контейнера MySQL 3306, а внешний 9000 к которому мы сможем подключиться через MySQL Workbench.

Указав в нашей службе api depends\_on: -db мы принудительно запускаем службу db первой перед api.

В обозревателе решений откроем файл docker-compose.override.yml (Рисунок 2.6)  
и изменим его следующим образом чтобы наши порты заработали (Рисунок 2.7).

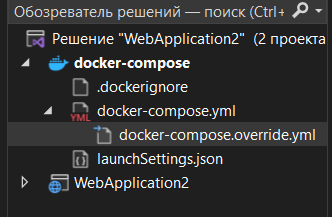


Рисунок 2.6 – Файл docker-compose.yml

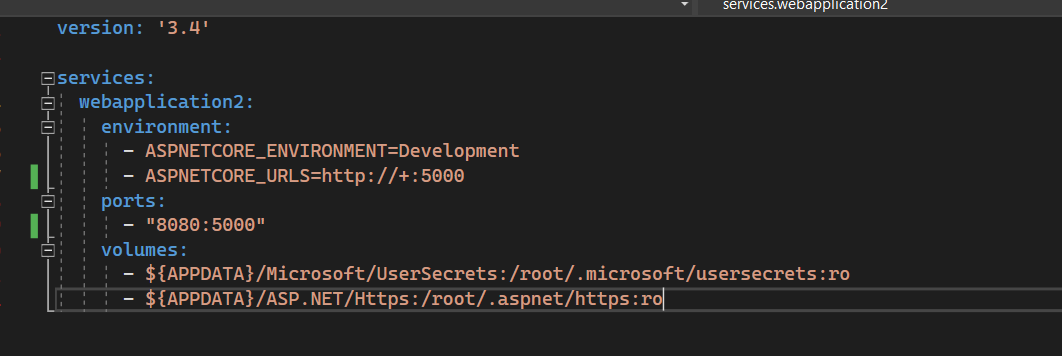
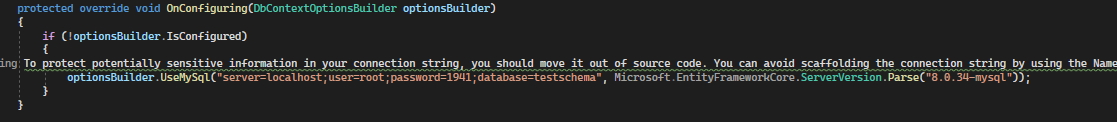


Рисунок 2.7 – Файл docker-compose.yml

ASPNETCORE\_URLS\_http://+:5000 – указывает на то что приложение будет прослушиваться на порту 5000 для всех ip-адресов

Осталось самое важное – поменять значения в строке подключения в нашем testschemaContext.cs который находится в папке Model (Рисунок 2.8)

Рисунок 2.8 – Файл testschemaContext.cs

В server мы должны ввести имя нашего контейнера, которое мы ему дали, когда добавляли службу db. Поля user и password заполняются аналогичным образом теми данными, которые мы указали для запуска службы db (Рисунок 2.9).

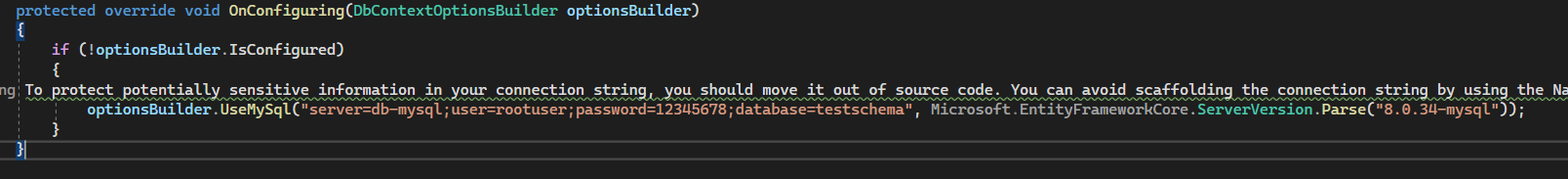


Рисунок 2.9 – Файл testschemaContext.cs с измененной строкой подключения

# Запуск docker-compose

Для запуска мы можем использовать и встроенное средство, но мы будем использовать командную строку. Перейдем в папку проекта командой cd (путь к папке): cd Users\Дмитрий\sourse\repos\WebApplication2. Далее введем команду **docker-compose up --build.**

Как только все будет запущено мы сможем перейти в <http://localhost:8080/Swagger>

либо сразу сделать запрос <http://localhost:8080/User> и получить данные пользователей.