Инструкция по подготовке среды разработки:

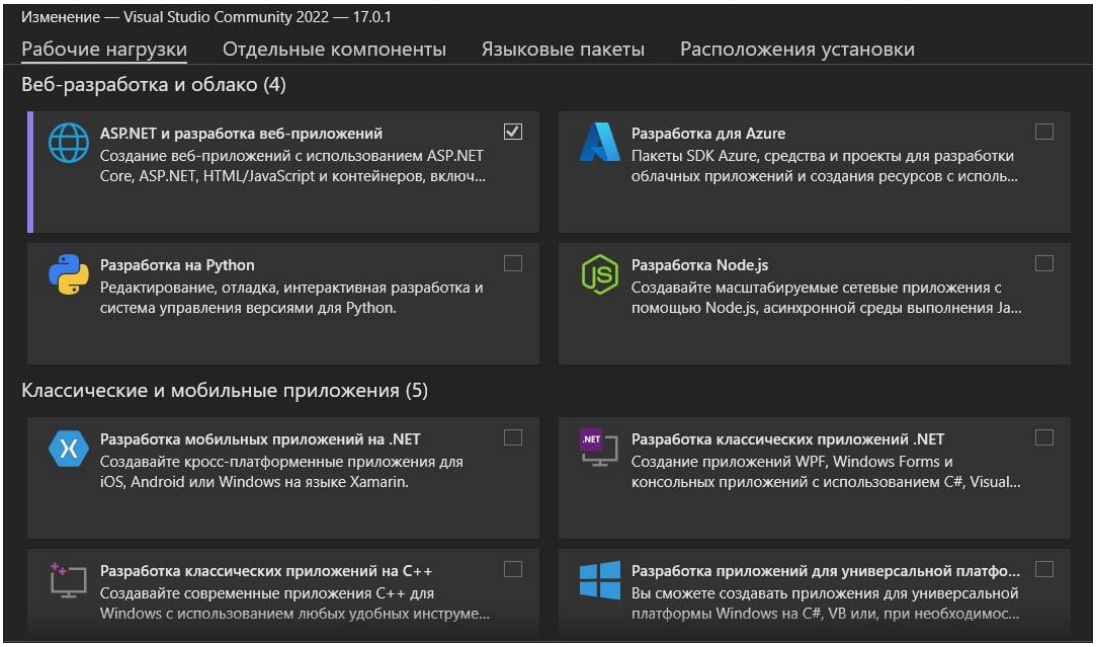
PostgreSQL v12 и выше
Visual Studio IDE 2022
Для удобства можно использовать DataGrip или другие IDE для работы с БД

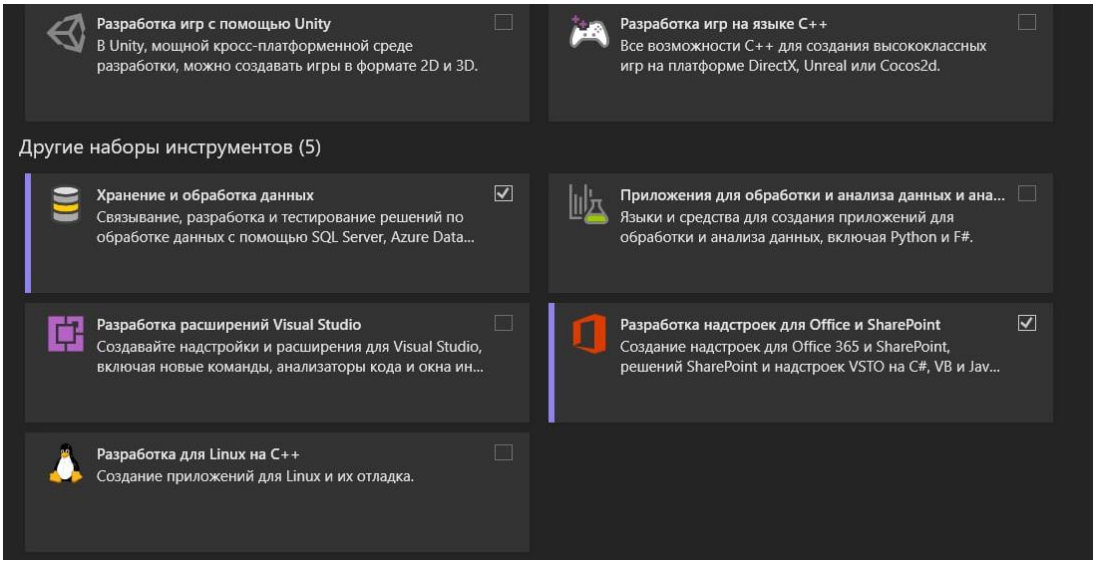
Ссылки:

PostgreSQL - https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
VS 2022 - https://visualstudio.microsoft.com/ru/launch/
DataGrip - https://www.jetbrains.com/datagrip/

1 шаг:

Ставим VS 2022 Community Edition, выбираем в Рабочих Нагрузках нагрузки "ASP.NET и разработка веб-приложений", "Хранение и обработка данных" и "Разработка надстроек для Office и SharePoint".





2 шаг:

Ставим PostgreSQL, в установщике выбираем все по дефолту. Пытаемся через pgAdmin зайти в сервер. Если сервер не запущен - запускаем через cmd:

psql start -D <путь к базе>

Если сервер запустился и отображается в pgAdmin - все выполнено правильно.

3 шаг:

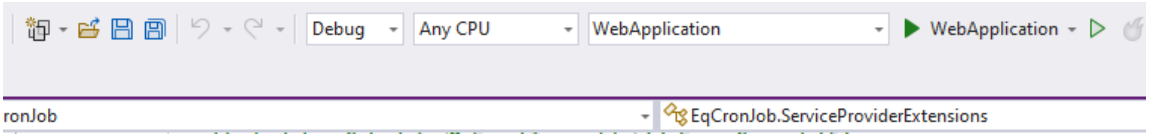
Ставим DataGrip, в установке ставим все по дефолту.
4 шаг:

В самой DataGrip нажимаем на "+", выбираем Add Data Source > PostgreSQL > в параметрах указываем:
Host: localhost
Port: 5432
User, Database и Password заполняем так, как выбирали пароль и логин при установке
Нажимаем Test Connection, тестируем подключение. Если все нормально - нажимаем OK, подключаемся к базе.
5 шаг:
Чтобы выгрузить к себе базу, подключаемся аналогично к базе eq_aggregator_dev

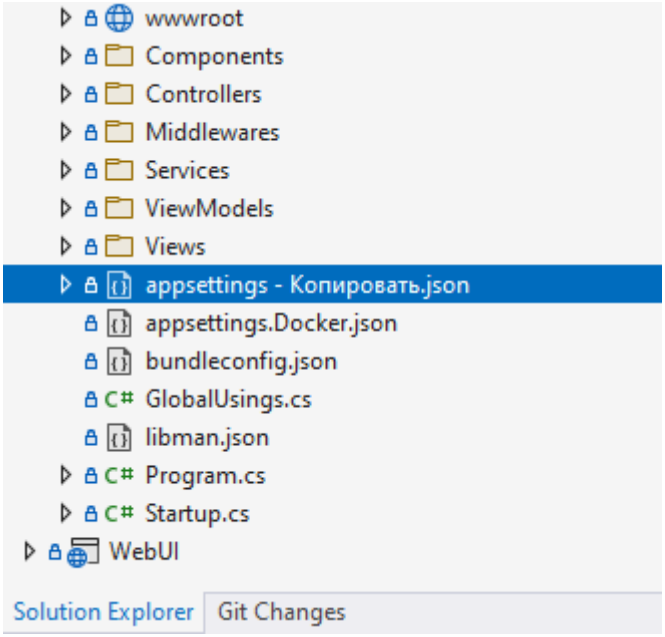
Host: 77.39.15.141
Port: 9595
Username: postgres
Password: 1q2w3e4r
Database: eq_aggregator_dev

После подключения в DataGrip выбираем базу, нажимаем ПКМ, выбираем экспортировать через pg_dump. Указываем pg_dump.exe файл там, куда устанавливали PostgreSQL, затем указываем путь, куда выгрузим дамп. В базу localhost импортируем также > ПКМ > "Restore with psql" > выбираем psql.exe в директории установки PostgreSQL > выбираем путь к дампу > выгружаем. (ВАЖНО: База

localhost должна иметь кодировку UTF-8, иначе могут возникнуть проблемы с импортом!)
6 шаг:
Как только получили доступ к репозиторию eq.Aggregator на GitHub, заходим в Visual Studio 2022, выбираем "Клонировать репозиторий", берем ссылку на GitHub (https://github.com/murtazo96/-eq-.Aggregator), вставляем в VS, клонируем репозиторий. Ставим проект по умолчанию "WebApplication". IIS Express меняем на WebApplication.



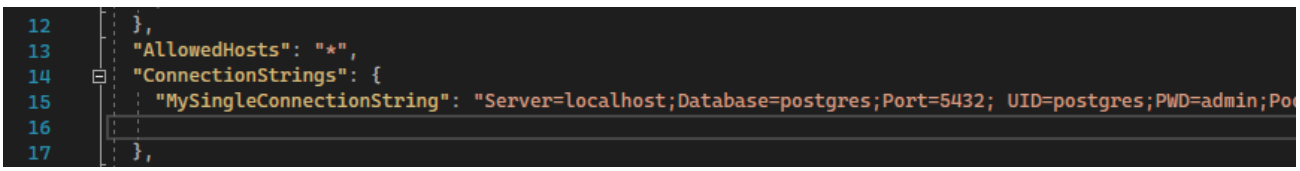
После этого создаем новый файл appsettings.json в корне проекта WebApplication и копируем в него содержимое appsettings - Копировать..json



В файле appsettings.json:

Заменить строку подключения к бд, параметр MySingleConnectionString. В этой строке пишем параметры нашей базы localhost.

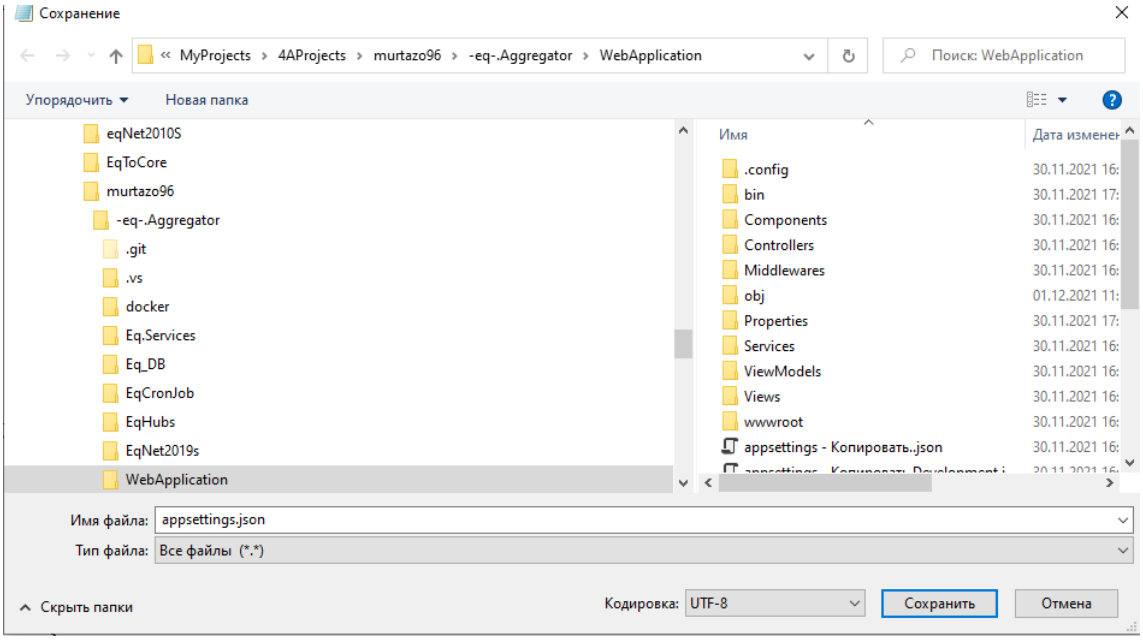
Server - IP хоста, в нашем случае localhost
Database - имя локальной БД
Port - порт хоста
UID - имя пользователя локальной БД
PWD - пароль от пользователя (если был задан)
Все остальное оставляем по умолчанию.



Пример как должна выглядеть MySingleConnectionString

7 шаг:

Заходим в папку, где находится наш файл appsettings.json, выбираем "Открыть с помощью блокнота", во вкладке Файл выбираем "Сохранить как", там ставим "All files", имя файла должно быть appsettings.json. Кодировку внизу выбираем UTF-8! Сохраняем, перезаписываем файл.



После всех этих шагов мы можем запустить дебаг и проверить приложение. По умолчанию оно развертывается на localhost:5151.

Дополнительно:

Для дополнительного удобства можно удалить или закомментировать раздел Kestrel в файле appsettings.json. Тогда можно будет поставить IIS Express вместо WebApplication в дебаге, и приложение будет автоматически открываться в браузере

Для пользователей Ubuntu (скопировано с GitHub):

Только для Ubuntu 16.04, 18.04, 20.04!

1. Опубликовать приложение для linux-x64 и скопировать на сервер.
2. Установить nginx

sudo -s nginx=stable # use nginx=development for latest development version

add-apt-repository ppa:nginx/$nginx

apt-get update

apt-get install nginx

3. Перезапустить и проверить правильно ли установлен nginx

```
sudo service nginx start
```

4. Конфигурировать nginx как обратный прокси для asp.net core приложения
4.1. Создать или изменить файл /etc/nginx/sites-available/default:

```
      server {
    listen 80;
    location / {
    proxy_pass http://localhost:5000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection keep-alive;
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
} }
```

Порты заменить на используемые(80 и 5000 заменить если требуется)

4.2. Проверить конфигурацию nginx на правильность:

```
nginx -t
```

4.3. Перезапустить

```
sudo nginx -s reload
```

4.4. Create symbolic link:

```
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

5. Перейти в папку дистрибутив и убедиться что приложение запускается без ошибок:

```
dotnet WebApplication.dll
```

6. Присвоить permisions если потребуется:

```
chmod u+x WebApplication.dll
```

7. Теперь если приложение запускается и не крашится,нужно создать сервис (например с названием EqToCore или что-то другое) , который всегда запускается приложение при запуске системы или когда оно крашится:

```
sudo nano /etc/systemd/system/EqToCore.service
```

8. Внутри разместить:

```
[Unit]
Description=My first .NET Core application on Ubuntu
```

```
[Service]
WorkingDirectory=/home/test/default_publish
ExecStart=/usr/bin/dotnet /home/test/default_publish/WebApplication.dll
Restart=always
RestartSec=10 # Restart service after 10 seconds if dotnet service crashes
SyslogIdentifier=offershare-web-app
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

9. Изменить путь к папке приложения, ASPNETCORE_ENVIRONMENT и другие параметры(по желанию)

10. Включить сервис:

```
sudo systemctl enable EqToCore.service
```

11. Запустить сервис:

```
sudo systemctl start EqToCore.service
```

12. Проверить статус:

```
sudo systemctl status EqToCore.service
```

13. Если сервис активен и по заданному ip не открывается сайт, то перезапустить nginx:

```
sudo systemctl restart nginx
```