**СОДЕРЖАНИЕ**

[Введение 3](#_Toc171431626)

[1 Установка и настройка ПО 4](#_Toc171431627)

[1.1 Настройка Git 4](#_Toc171431628)

[1.2 Установка npm и Angular 5](#_Toc171431629)

[1.3 Установка VS Code 6](#_Toc171431630)

[1.4 Настройка VS Code для работы с Angular+.Net Core 7](#_Toc171431631)

[2 Начало работы 9](#_Toc171431632)

[2.1 Создание бота в Telegram 9](#_Toc171431633)

[2.2 Создание команд для бота в Telegram 10](#_Toc171431634)

[2.3 Регистрируем google аккаунт 10](#_Toc171431635)

[2.4 Создаем гугл таблицу (ГТ) 12](#_Toc171431636)

[2.5 Пишем скрипт 13](#_Toc171431637)

[2.6 Библиотека 15](#_Toc171431638)

[2.7 Начало работы с кодом 16](#_Toc171431639)

[2.8 Парсинг страниц 17](#_Toc171431640)

[2.9 Публикация Angular приложения на Pages в GitHub 17](#_Toc171431641)

[2.10 Работа с dotnet 18](#_Toc171431642)

[2.11 Работа с Nuget 18](#_Toc171431643)

[2.12 Создаем ASP.NET Core Web API (C#) 20](#_Toc171431644)

[2.13 Создаем Angular веб-приложение 22](#_Toc171431645)

[3 Публикация 26](#_Toc171431646)

# Введение

Краткое пояснение как работать в связке Frontend (Angular) + Backend (Web API на .Net Core) с использованием VS Code под ОС Windows.

# Установка и настройка ПО

## Настройка Git

Ставим версию с оф.сайта <https://git-scm.com/download/win>

Запускаем консоль гита и заполняем имя пользователя и email

git config --global user.name "Chebatz"

git config --global user.email "chebatz@mail.ru"

Переходим в директорию с проектом

cd d:/10cip/work/prokrasotu

Создаем в этой папке локальный репозиторий

git init

Регистрируемся на <https://github.com/> и создаем внешний репозиторий под проект. Должны получить URL нашего репозитория: <https://github.com/chebatz/prokrasotu.git>

Связываем локальный и внешний репозиторий

git remote add origin https://github.com/chebatz/prokrasotu.git

Скачать себе копию в локальный репозиторий, сначала перейти в папку, затем выполнить

git clone <https://github.com/chebatz/prokrasotu.git>

Проверить что видит git, в каких файлах были изменения

git status

Добавить файл для отслеживания:

git add Readme.docx

Коммит на сервер:

git commit -m "added readme.docx"

Посмотреть логи последних действий:

git log

После изменений снова надо делать add и commit.

Чтобы загрузить файлы, которые ранее были сохранены через комит, надо узнать кеш (по git log) и выполнить:

git checkout b84204fb8a8de9d466a29d98ea58edb078d60a7a

Чтобы отправить все свои локальный комиты на github - выполнить:

git push origin master

git push --set-upstream origin main

Чтобы скачать себе комиты в локальный репозиторий с github - выполнить:

git pull origin master

Работа с ветками

Посмотреть ветки:

git branch

Создать ветку «feature»:

git branch feature

Переключиться на ветку «feature»:

git checkout feature

Отправить ветку «feature» на github:

git push origin feature

Перенести изменения из ветки «feature» в основную ветку через слияние.

Первый механизм через merge

Сначала делаем чекаут на ту ветку в котороую делаем merge

git checkout master

Затем сливаем все комиты в виде одного на ветку master. Изменения попадут в master (локальный репозиторий) одним комитом.

git merge feature

Второй механизм через **git rebase**

Перенесет все комиты (а не одним) и сохранит историю комитов.

## Установка npm и Angular

В Windows сначала установить Nodejs, после этого в Code появится npm.

Если требуется настроить прокси, то создать файл C:\Users\<username>\.npmrc следующего содержания:

proxy=http://<username>:<pass>@proxyhost:<port>

https-proxy=http://<uname>:<pass>@proxyhost:<port>

Символы заменять URL кодированием, например

\ == %5C

! == %21

proxyhost – это ip.

В терминале Linux: sudo dnf install npm

Затем:

npm uninstall -g @angular/cli

npm install -g @angular/cli@latest

Если выходит ошибка доступа EACCESS в логах npm, при которой не создаются папки:

Создаем директорию: mkdir ~/.npm-global

Конфигурируем npm на использование: npm config set prefix '~/.npm-global'

В домашней директории открываем или создаем ~/.profile файл и добавляем в него строку: export PATH=~/.npm-global/bin:$PATH

Обновляем системные переменные: source ~/.profile

После этого Angular должен установиться.

## Установка VS Code

Сначала надо поставить VS Code.

Выполнил в терминале dnf check-update и пакеты обновились. В том числе VS Code. Устанавливается все из локального репозитория. Для настройки локального репозитория обращаться в ОВИАСЗИР.

Если вылетает с ошибкой 407, то необходимо изменить файл **etc\dnf\dnf.conf**. Используем команду sudo vi dnf.conf из директории. Затем нажимаем i переходим в режим вставки и редактирований. Затем Esc и команду :wq для сохранения. Под этими данными будут работать все пользователи ноутбука.

Содержимое dnf.conf:

[main]

gpgcheck=1

installonly\_limit=3

clean\_requirements\_on\_remove=True

best=True

skip\_if\_unavailable=False

proxy=http://proxy.**snos.ru**:3128

proxy\_username=**ASU\**10cip

proxy\_password=SUPERSECRETPASSWORD

Теперь помни, что пароль тут надо обновлять.

Затем выполняем sudo dnf install code

## Настройка VS Code для работы с Angular+.Net Core

Для запуска VS Code в консоли набрать code.

Добавить расширения в VSCode (вкладка Extensions слева ):

* C#
* vscode-icons
* Visual Studio IntelliCode
* NuGet Package Manager
* VS Sharper for C#

**Работа с проектом C# через .NET CLI (command line interface) overview (**[**https://learn.microsoft.com/en-us/dotnet/core/tools/**](https://learn.microsoft.com/en-us/dotnet/core/tools/)**)**

Посмотреть в терминале в какой папке находишься: **PWD**

Всегда смотреть в какой директории находишься перед командой в терминале.

Если не ставятся расширения через магазин Code, то можно скачать и установить вручную с сайта <https://marketplace.visualstudio.com/>

Затем:

Go to the Extensions view.

Click Views and More Actions...

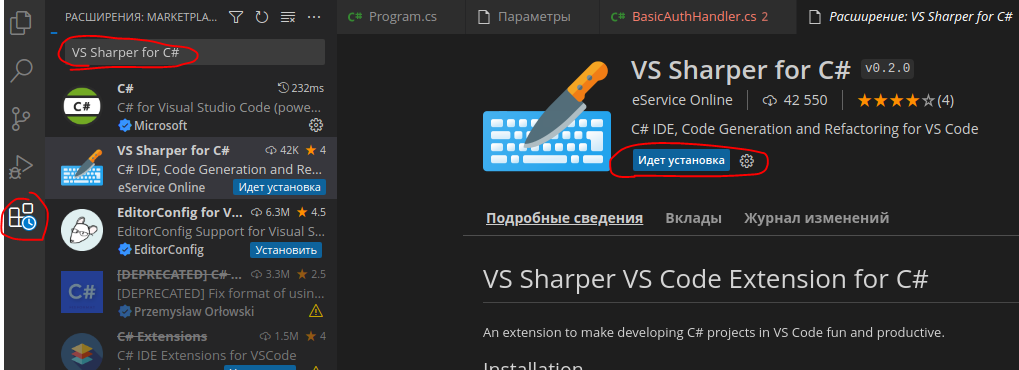
Select Install from VSIX...

или

# if you use VS Code

code --install-extension my-extension-0.0.1.vsix

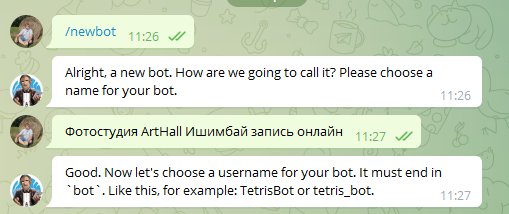
Чтобы при нажатии правой кнопкой появлялось меню «Новый C# класс» необходимо установить расширение VS Sharper for C#

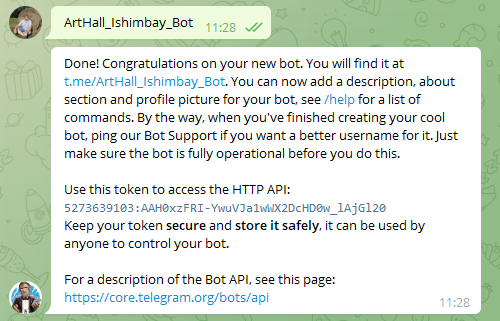


# Начало работы

## Создание бота в Telegram

Создаем бота через @BotFather в Telegram



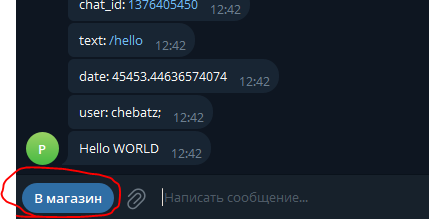


Получаем API-ключ

5273639103:AAH0xzFRI-YwuVJa1wWX2DcHD0w\_lAjGl20

Включаем inline mode через @BotFather

Кнопка для запуска приложения создается через команду /setmenubutton



Для того чтобы бот постоянно крутился в активном состоянии можно его запустить через Google Tables (Apps Scripts)

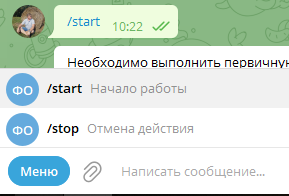
## Создание команд для бота в Telegram

Регистрируем команды в BotFather

*start - Начало работы*

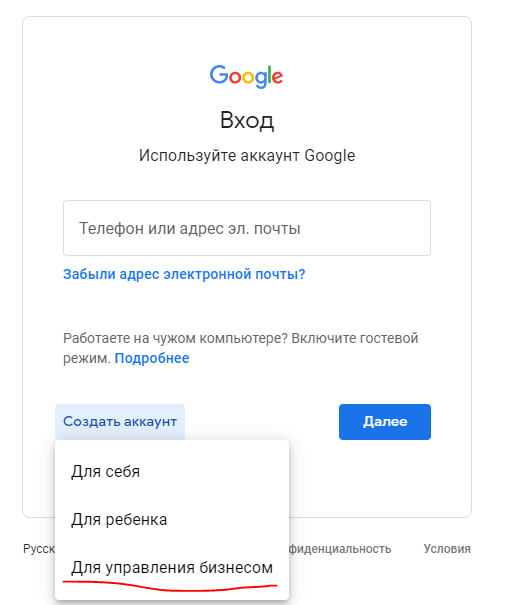
*stop - Отмена действия*

Итог выглядит так:



## Регистрируем google аккаунт

<https://accounts.google.com/signin/v2/identifier?hl=ru&continue=https%3A%2F%2Fwww.google.com%3Fhl%3Dru&ec=GAlA8wE&flowName=GlifWebSignIn&flowEntry=AddSession>

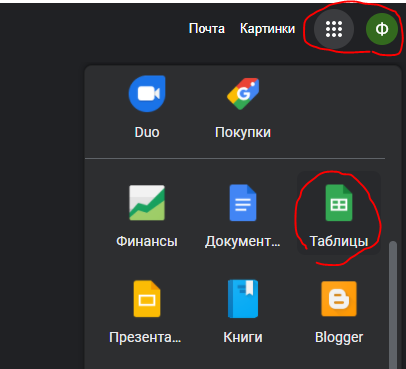


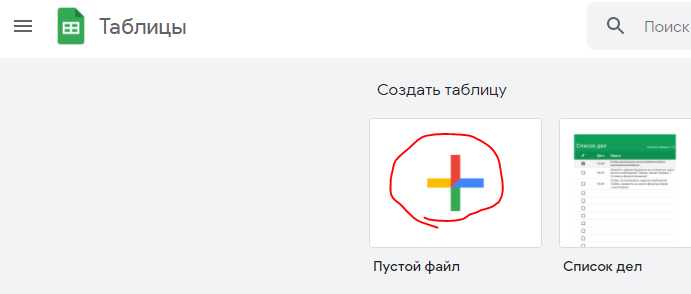
[arthall.ishimbay@gmail.com](mailto:arthall.ishimbay@gmail.com)

Если нужно – регистрируем компанию.

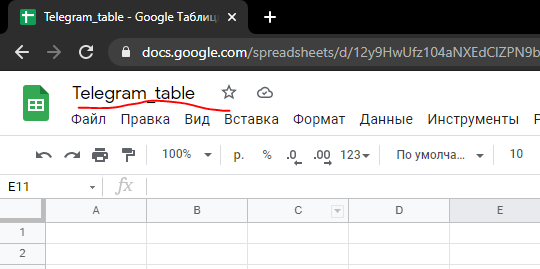
Дальше работаем от этой учетки.

## Создаем гугл таблицу (ГТ)



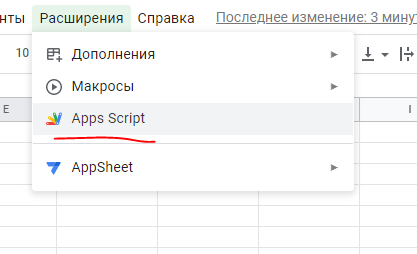


Называем таблицу



## Пишем скрипт

Заходим в редактор скриптов



function doPost(e) {

  var update = JSON.parse(e.postData.contents);

  var DOC = SpreadsheetApp.openById("12y9HwUfz104aNXEdClZPN9buHLaZpBKZOiHbYNaRFH4");

  //нам нужен только тип сообщение

  if(update.hasOwnProperty("message"))

  {

    var msg = update.message;

    var chat\_id = msg.chat.id;

    var text = msg.text;

    var msg\_array = msg.split(" ");

    var date = (msg.date/86400)+255569.125;

    var user = msg.from.username;

    if (msg\_array[0]=="/hello")

    {

      send("Hello WORLD", chat\_id);

    }

  }

}

function send(msg, chat\_id)

{

  //отправляет сообщения в тлг. на вход функции дать сообщение и ID чата, в который нужно провести отправку

  var payload={

    "method":"sendMessage",

    "chat\_id":String(chat\_id),

    "text":msg,

    "parse\_mode":"HTML"

  }

  var data = {

    "method":"post",

    "payload":payload

  }

  var API\_TOKEN = "5273639103:AAH0xzFRI-YwuVJa1wWX2DcHD0w\_lAjGl20";

  UrlFetchApp.fetch("https://api.telegram.org/bot"+API\_TOKEN+"/", data);

}

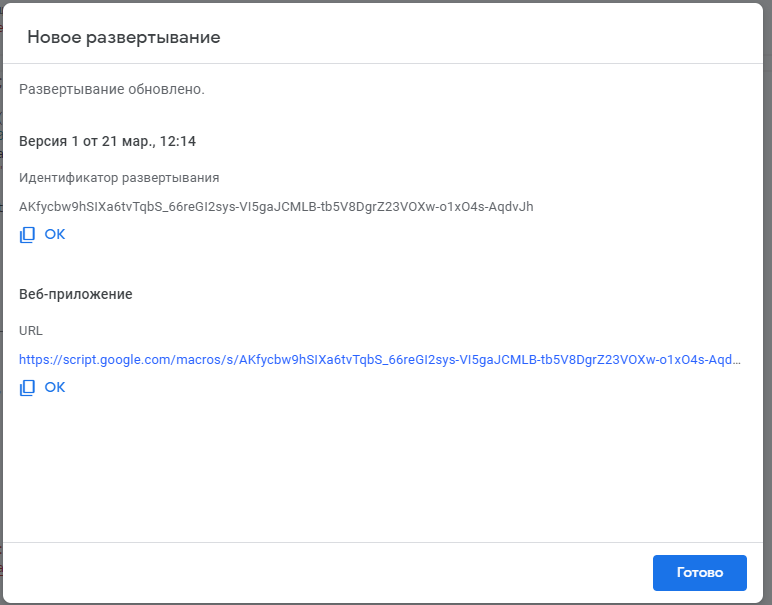
Регистрируем webHook:

Сначала разворачиваем проект



…как Web приложение, запуск «от моего имени», доступ «Все».

Разрешаем доступ…



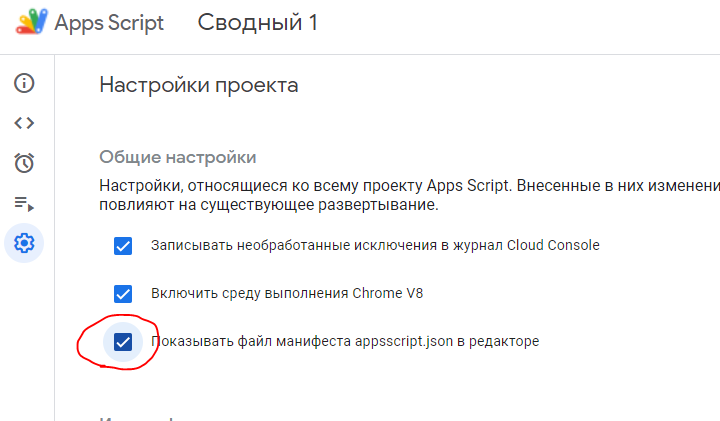
<https://api.telegram.org/bot5273639103:AAH0xzFRI-YwuVJa1wWX2DcHD0w_lAjGl20/setWebHook?url=https://script.google.com/macros/s/AKfycbw9hSIXa6tvTqbS_66reGI2sys-VI5gaJCMLB-tb5V8DgrZ23VOXw-o1xO4s-AqdvJh/exec>

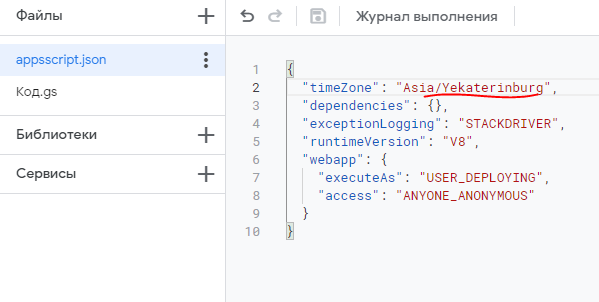
В результате выполнения получаем строку

{"ok":true,"result":true,"description":"Webhook was set"}

Значит все ОК.

Меняем TimeZone





## Библиотека

<https://habr.com/ru/articles/586744/>

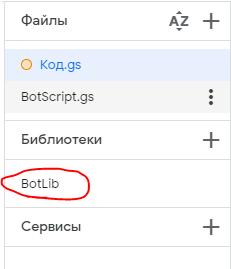
## Начало работы с кодом

Создаем таблицу и переходим в редактор Apps Script

Создаем файл BotScript.gs вставляем в него код. Развертываем его как библиотеку. Присваиваем имя библиотеки. Код берем из ссылки.

<https://script.google.com/macros/library/d/1-ss6TiF8rVbhsJhUCufP9kE40f2jfoL8-ml0sJbEl9lXkIgfUPMLTYtt/1>

Добавляем эту библиотеку в проект.



В файле код.gs прописываем код на запуск библиотеки

Запуск лучше делать в отдельном проекте (не глючит библиотека при компиляции), а в конце переподвязать на основной скрипт (перед тем как отдать в продуктив).

Один проект (основной) постоянно компилим как библиотеку, второй проект во втором файле один раз компилим как веб приложение и делаем на него веб хук. Используем в проекте функции из библиотеки. Так быстрее разработка.

Развертываем как веб-приложение, копируем App\_link после развертывания.

const API\_TOKEN = '5273639103:AAH0xzFRI-YwuVJa1wWX2DcHD0w\_lAjGl20';

const App\_link = "https://script.google.com/macros/s/AKfycbya1mRu\_HgFGgLMxfrnaFdrK4XpW7rfFdrRcOn8i7jdMkHmtiLYLbWY7RKA1E-l9hpF/exec";

function doPost(e) {

  BotLib.doPost(e);

}

function api\_connector(){

  UrlFetchApp.fetch('https://api.telegram.org/bot' + API\_TOKEN + '/setWebHook?url='+App\_link);

}

Затем запускаем функцию api\_connector, которая свяжет telegram бот со скриптом.

На Лист1 в таблице копируем настройки бота

## Парсинг страниц

Надо подключить библиотеку cheerio

1ReeQ6WO8kKNxoaA\_O0XEQ589cIrRvEBA9qcWpNqdOP17i47u6N9M5Xh0

const contentText = UrlFetchApp.fetch('https://www.somesite.com/').getContentText();

const $ = Cheerio.load(contentText);

$('.some-class').first().text();

## Публикация Angular приложения на Pages в GitHub

Чтобы приложение имело публичную ссылку, работало и можно было его привязать к телеграм боту через /setmenubutton в @BotFather.

Регистрируем аккаунт на GitHub и создаем проект.

Читаем и делаем как тут написано.

<https://dev.to/apdharshi/deploying-your-angular-application-to-github-pages-4laf>

Публикуем по

ng build --configuration production --output-hashing=none --base-href "https://ichebat.github.io/prokrasotu/"

git add .

git commit –m “Пишем коммент”

git push origin main

Если не работает, то надо в PowerShell изменить политику

Set-ExecutionPolicy -Scope CurrentUser RemoteSigned

Затем вернуть на значение по умолчанию:

Set-ExecutionPolicy -Scope CurrentUser Undefined

Проверка списком

Get-ExecutionPolicy -List

Затем деплой на ГитХаб по команде

angular-cli-ghpages --dir=dist/angular/browser

В результате получаем работающую ссылку, которую можно привязать к боту как приложение.

<https://ichebat.github.io/prokrasotu/>

## Работа с dotnet

Создать sln файл проекта в выбранной директории: выполнить в терминале

dotnet new sln –n “PIWebAppSln”

Создать проект типа консольное приложение

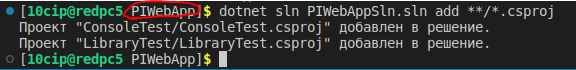
dotnet new console –n “ConsoleTest”

Создать проект типа библиотека

dotnet new classlib –n “LibraryTest”

Добавить оба проекта в sln

dotnet sln PIWebAppSln.sln add \*\*/\*.csproj



Добавить в проект **ConsoleTest** ссылку на проект **LibraryTest**

dotnet add ConsoleTest/ConsoleTest.csproj reference LibraryTest/LibraryTest.csproj



Для начала работы с **проектом** необходимо перейти в **папку проекта**.

cd ConsoleTest

## Работа с Google Sheets как с JSON

Сначала надо опубликовать лист и получить ссылку. Чтобы скачать данные с листа как JSON надо подставить в ссылку SheetId (Id книги), Gig (Id листа). Потом будет работать Get.

sheetId: '1JPSzoAEUktlPgShanrrdZs3Vb5YwQVzlTeog8JmzWrI',

sheetProductsGid: '1383014775',

getProductsFromGoogleAsJSONUrl: 'https://docs.google.com/spreadsheets/d/1JPSzoAEUktlPgShanrrdZs3Vb5YwQVzlTeog8JmzWrI/gviz/tq?tqx=out:json&tq&gid=1383014775',

getDelivery(): Observable<IDelivery[]> {

console.log('Start get delivery');

return this.\_http.get(this.urlDelivery, { responseType: 'text' }).pipe(

map((res: any) => {

let gsDataJSON = JSON.parse(res.substring(47, res.length - 2));

console.log(gsDataJSON);

return gsDataJSON.table.rows.map(function (row: any): IDelivery {

return {

id: row.c[0] ? row.c[0].v : '',

name: row.c[1] ? row.c[1].v : '',

description: row.c[2] ? row.c[2].v : '',

amount: row.c[3] ? row.c[3].v : '',

freeAmount: row.c[4] ? row.c[4].v : '',

};

});

}),

);

}

## Работа с Nuget

**NuGet** необходимо настроить для работы через proxy

**Для винды:**

Скачиваем nuget.exe <https://www.nuget.org/downloads>

Выполняем команды:

nuget.exe config -set http\_proxy=http://10.1.3.41:3128

nuget.exe config -set http\_proxy.user=asu\10cip

nuget.exe config -set http\_proxy.password=!superpassword!

Команды создают NuGet.config файл d:\Users\10cip\AppData\Roaming\NuGet\

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<packageSources>

<add key="nuget.org" value="https://api.nuget.org/v3/index.json" protocolVersion="3" />

</packageSources>

<disabledPackageSources>

<add key="Microsoft and .NET" value="true" />

</disabledPackageSources>

<config>

<add key="http\_proxy" value="http://10.1.3.41:3128/" />

<add key="http\_proxy.user" value="asu\10cip" />

<add key="http\_proxy.password" value="AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAI0+LlArSw0ieXgO3tybcfQAAAAACAAAAAAADZgAAwAAAABAAAAB1hqgSY+V4ONnz59yVqBFfAAAAAASAAACgAAAAEAAAAIPIyE7apmL8hrrVxc/9P2MYAAAAIhwZ2YveVq4sINMqAiTeJJsDw/3sNNllFAAAAOZpISSONVfRgn5WcnI0F/TUqrp3" />

</config>

</configuration>

После этого все работает.

**Для Линукс:**

**Если линуксовая учетка в группе proxyusers на corp.lan то проблем быть не должно… иначе не получилось завести NuGet**

Проверить доступность интернета через терминал

wget ya.ru

Экспорт настроек прокси через терминал

export http\_proxy=http://ASU\10cip:ПАРОЛЬ@10.1.3.41:3128/

export http\_proxy=http://10cip:123qweQWE@proxy.corp.lan:3128/

Сначала поставить gssntlmssp <https://centos.pkgs.org/8/centos-appstream-x86_64/gssntlmssp-0.7.0-6.el8.x86_64.rpm.html>

dnf install gssntlmssp

Чистка кеша NuGet

sudo dotnet nuget locals -c all

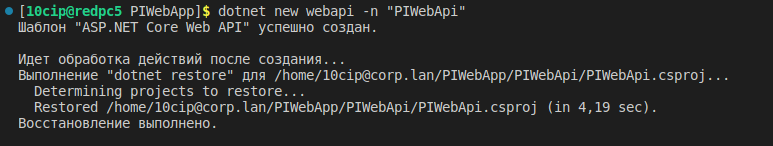
dotnet nuget add source "https://api.nuget.org/v3/index.json" --name "nuget.org" -- http\_proxy "http://10.1.3.41:3128/" --http\_proxy.user "asu\10cip" --http\_proxy.password "PASSWORD!" --configfile ~/.nuget/NuGet/NuGet.Config

Для добавление пакета **Dapper** через NuGet перейти в терминале в папку с проектом

dotnet add package Dapper

## Создаем ASP.NET Core Web API (C#)

dotnet new webapi -n "PIWebApi"



Перейти в папку cd PIWebApi и выполнить dotnet restore

Для запуска dotnet run

[**https://www.youtube.com/watch?v=S5dzfuh3t8U**](https://www.youtube.com/watch?v=S5dzfuh3t8U)

Ставить под .net7.0

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools

Чтобы заработал scaffolding при создании файлов (а не с нуля заполнять все содержимое каждого .cs файла)

dotnet tool install -g dotnet-aspnet-codegenerator

Может потребоваться также:

dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design   
dotnet add package Microsoft.EntityFrameworkCore.Design   
dotnet add package Microsoft.EntityFrameworkCore.SqlServer

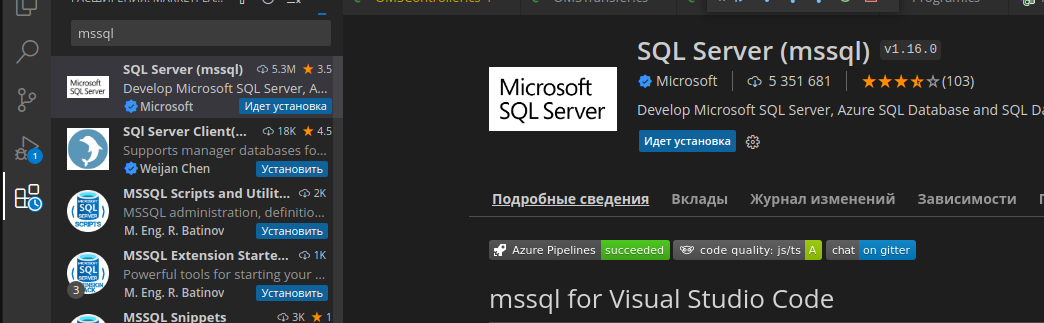
помощь

dotnet aspnet-codegenerator controller –h

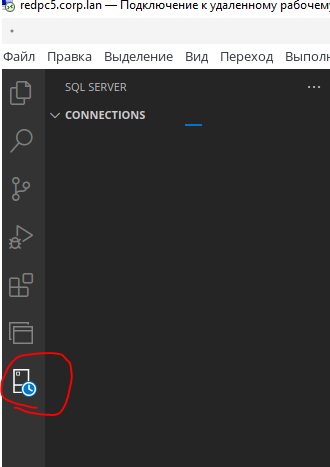
To scaffolder a controller, we need to use the command below:  
dotnet aspnet-codegenerator controller -name OMSController --relativeFolderPath Controllers --useDefaultLayout --referenceScriptLibraries --restWithNoViews --useAsyncActions --readWriteActions

To scaffolder a view, the command is like below:  
dotnet aspnet-codegenerator view Index List -m Book -outDir Views\Books --useDefaultLayout --referenceScriptLibraries

Для подключения к sql server



Появится значок



Дебажить через

dotnet run --arch x64 --os linux

Если не заработал дебаг по F5 под Linux в VS Code в Web API .Net CORE

Добавить в .csproj строки:

<Target Name="PostBuild" AfterTargets="PostBuildEvent">

<Exec Command="cp $(OutDir)runtimes/unix/lib/netcoreapp3.1/Microsoft.Data.SqlClient.dll $(OutDir)" />

</Target>

## Создаем Angular веб-приложение

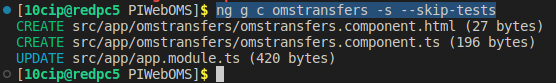
Создать новый Angular проект

ng new PIWebOMS --no-standalone

Если не работает то <https://stackoverflow.com/questions/65908987/how-to-open-visual-studio-codes-settings-json-file> и <https://stackoverflow.com/questions/56199111/visual-studio-code-cmd-error-cannot-be-loaded-because-running-scripts-is-disabl/67420296#67420296>

Добавить компонент oms-transfers в проект Angular

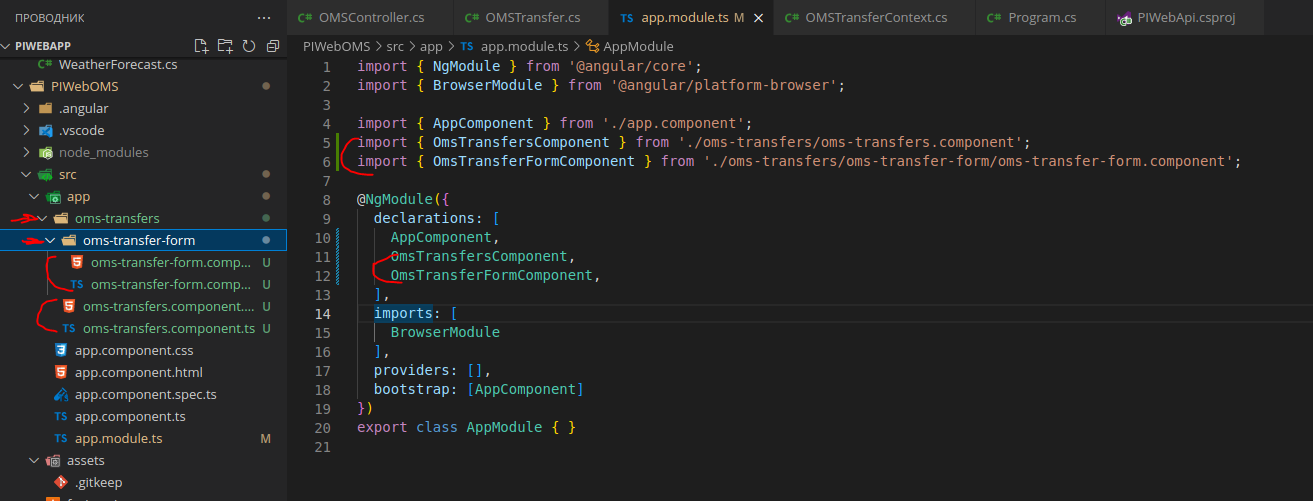
ng g c oms-transfers -s --skip-tests



Добавить вложенный компонент oms-transfer-form в проект Angular

ng g c oms-transfers/oms-transfer-form -s --skip-tests

AppModule апдэйтится автоматически



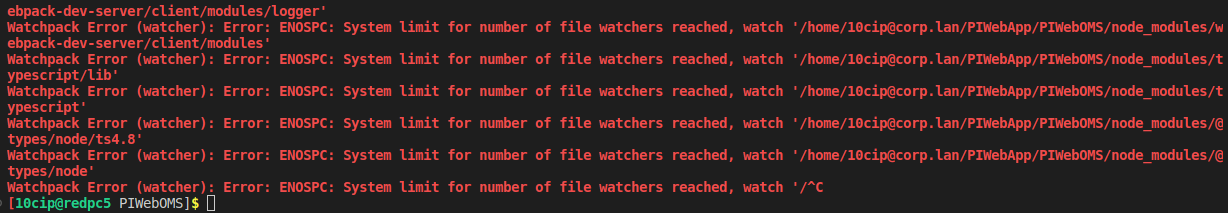
Добавить сервис (для связи с WebAPI) oms-transfer в проект Angular

ng g s shared/oms-transfer --skip-tests

Добавить класс для модели oms-transfer.model в проект Angular

ng g class shared/oms-transfer --type=model --skip-tests

Если ошибки



Выполнить

echo fs.inotify.max\_user\_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl –p

или удалить папку **node\_modules** и потом выполнить npm install

Установка bootstrap (<https://getbootstrap.com/>)

<https://www.freecodecamp.org/news/how-to-add-bootstrap-css-framework-to-an-angular-application/>

npm install bootstrap bootstrap-icons

Затем добавить в angular.json

"styles": [

"node\_modules/bootstrap/scss/bootstrap.scss",

"node\_modules/bootstrap-icons/font/bootstrap-icons.css",

"src/styles.scss"

],

"scripts": [

"node\_modules/bootstrap/dist/js/bootstrap.bundle.min.js"

]

Устанавливаем ng-bootstrab

ng add @ng-bootstrap/ng-bootstrap

src/app/app.component.ts добавляем

import { Component } from '@angular/core';

import { NgbModal } from '@ng-bootstrap/ng-bootstrap';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.scss'],

})

export class AppComponent {

constructor(private modalService: NgbModal) {

}

public open(modal: any): void {

this.modalService.open(modal);

}

}

Добавление GUID в Typescript

npm i guid-typescript --save

и

import { Guid } from 'guid-typescript';

export class GuidExample {

public id: Guid;

constructor() {

this.id = Guid.create(); // ==> b77d409a-10cd-4a47-8e94-b0cd0ab50aa1

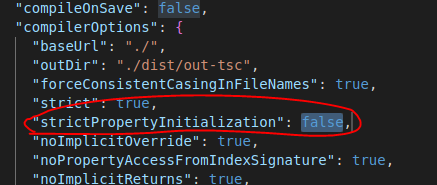
}

}

Чтобы убрать ошибки типа

Свойство "transferUID" не имеет инициализатора, и ему не гарантировано присваивание в конструкторе.ts(2564)

В файле tsconfig.json добавить строку



Не забыть включить CORS на сервере в API

builder.Services.AddCors();

Тему можно сделать на <http://mcg.mbitson.com/>

# Публикация

Развертывание web api под IIS если возникает ошибка Ошибка HTTP 500.19

**dotnet-hosting-2.2.1-win.exe (Хостинг бандл) и rewrite\_amd64\_ru-RU.msi (УРЛ рерайт)**

Сначала установить сервер nginx (

<https://medium.com/@technicadil_001/deploy-an-angular-app-with-nginx-a79cc1a44b49>

<https://docs.rackspace.com/support/how-to/install-nginx-on-fedora/>

<https://infoit.com.ua/linux/fedora/kak-ustanovit-nginx-na-fedora-35-34-33-32-31/>)

sudo dnf install nginx

меняем в своей папке конфиги и

cp [/home/10cip@corp.lan/nginx.conf /etc/nginx/nginx.conf](mailto:/home/10cip@corp.lan/nginx.conf%20/etc/nginx/nginx.conf)

В конфиге добавляем сервер

*server {*

*listen 9090;*

*server\_name localhost;*

*location / {*

*root html/piweb-oms;*

*index index.html index.htm;*

*try\_files $uri $uri/ /index.html;*

*}*

*error\_page 500 502 503 504 /50x.html;*

*location = /50x.html {*

*root html;*

*}*

*}*

Конфиг лежит в /etc/nginx/nginx.conf

Папка html лежит в /usr/share/nginx/html

Для Api настраиваем реверс прокси на порт 5000 (<https://setu677.medium.com/how-to-host-asp-net-core-on-linux-using-nginx-85339560e929> )

*server {*

*listen 9191;*

*server\_name localhost;*

*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;*

*}*

*}*

**ng build --configuration production --output-hashing=none --base-href “/”**

**и затем**

**копируем публикацию (получаем рут в консоли sudo su)**

yes | cp -r /home/10cip@corp.lan/PIWebApp/PIWebOMS/dist/piweb-oms /usr/share/nginx/html

Читаем <https://dev.to/apdharshi/deploying-your-angular-application-to-github-pages-4laf>

**dotnet publish --configuration Release**

**создается папка net6.0 c файлами, переименовываем**

mv /home/10cip@corp.lan/PIWebApp/PIWebApi/bin/Release/net6.0 /home/10cip@corp.lan/PIWebApp/PIWebApi/bin/Release/piweb-api-oms

**и затем**

**копируем публикацию**

yes | cp -r /home/10cip@corp.lan/PIWebApp/PIWebApi/bin/Release/piweb-api-oms /usr/share/nginx/html

**затем запускам web api на nginx в терминале из папки «**/usr/share/nginx/html»

**dotnet PIWebApi.dll**

**.**

**.**

**.**

**.**

**nginx команды**

sudo systemctl start nginx

sudo systemctl enable nginx

sudo systemctl status nginx

Если логи после публикации не пишутся, то надо создать папку с логами и дать на нее права.

sudo chmod –R 777 /usr/share/nginx/html/oms-viewer-api/logs/

Если nginx не работает на указанном порту, возможно он закрыт

<https://9to5answer.com/nodejs-nginx-error-13-permission-denied-while-connecting-to-upstream>