

¿Qué es?

En este documento se exponen los pasos a seguir para poder llegar al punto en el que poder ponerte a desarrollar módulos en C# para FiveM en Windows 10.

Pasos

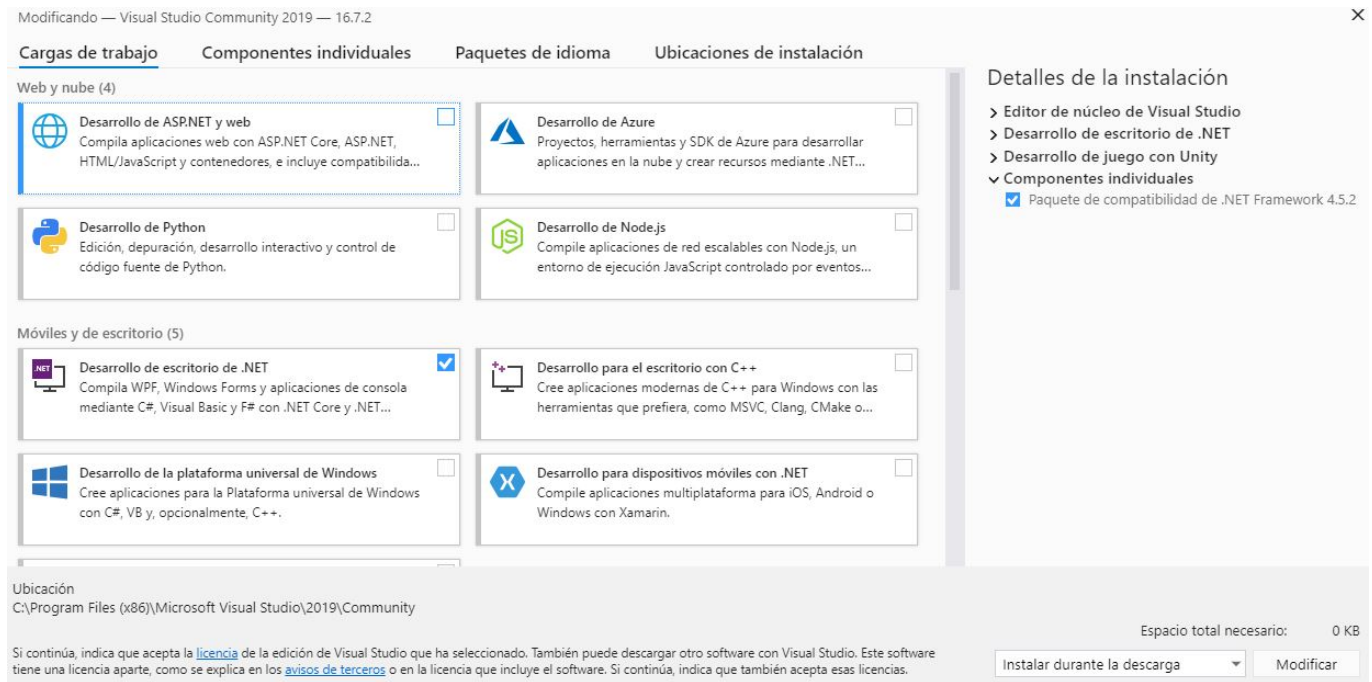
Por supuesto, tener comprado e instalado el GTA V, ya que cuando instalemos FiveM, nos pedirá la ruta de instalación del GTA V.

Instalación de Microsoft Visual Studio Community 2019

Este entorno nos aportará las herramientas para poder desarrollar en C#. La página web desde la que se puede descargar el software es [ésta](#), donde seleccionaremos **Community 2019** en el desplegable:



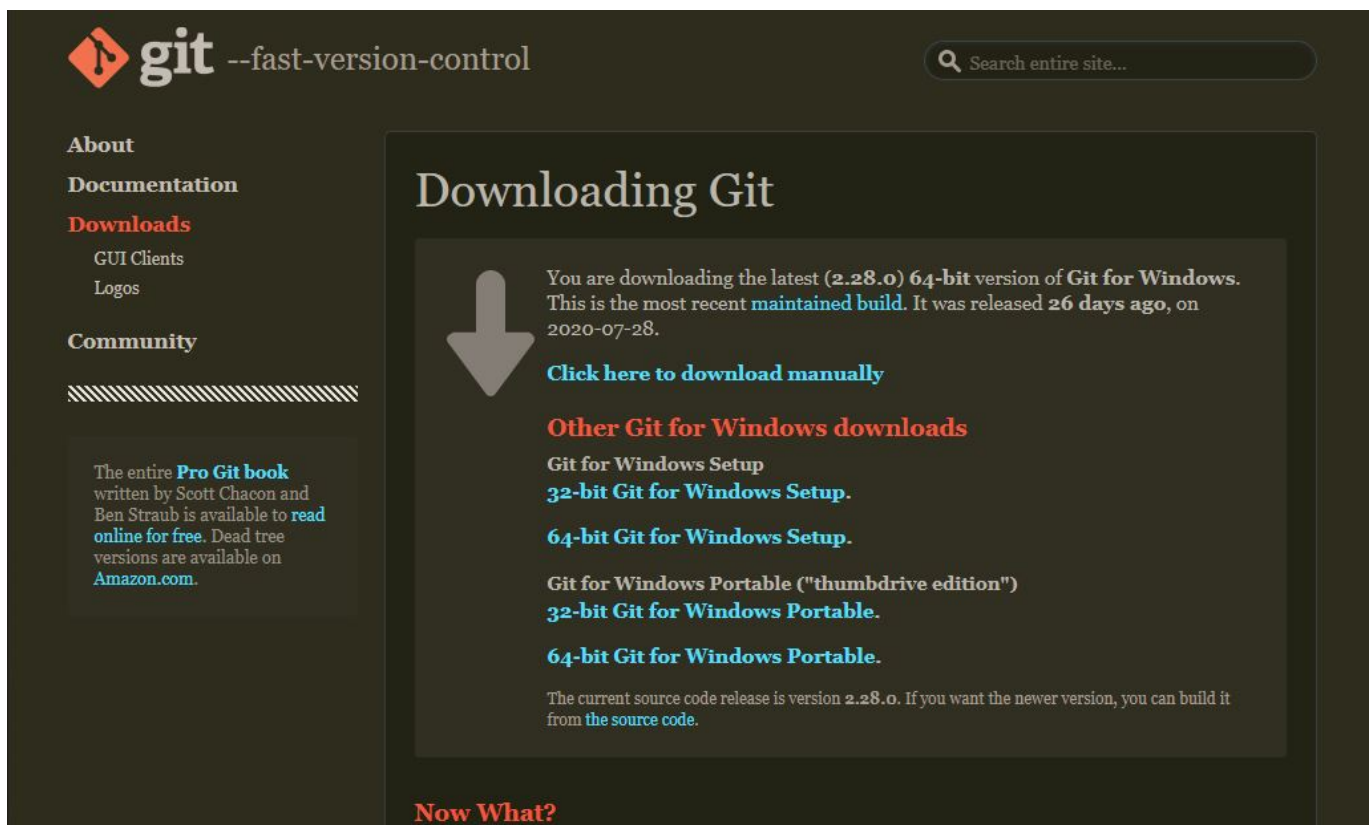
Terminada la descarga, procedemos a su instalación, donde deberemos seleccionar la opción **Desarrollo de escritorio de .NET** y avanzar con la instalación:



Cuando termine, continuar con el siguiente apartado. Más adelante indicaremos los pasos a seguir para crear un nuevo proyecto

Instalación de Git

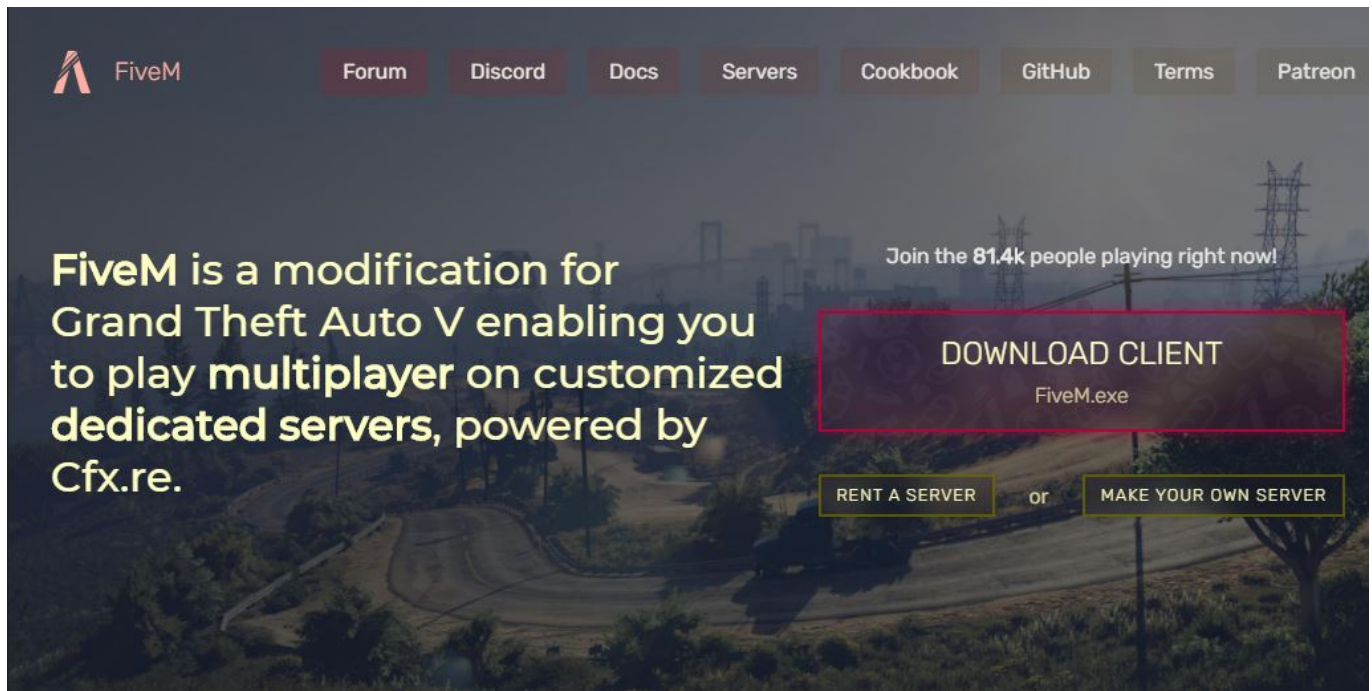
Nos hará falta para clonar repositorios e incluso, cuando desarrollemos, para poder compartir nuestro código con el mundo. Para ello vamos [aquí](#) y lo descargamos:



La instalación es sencilla y pueden dejarse los parámetros por defecto, al menos que se desee modificar alguno.

Instalación de FiveM

Para poder desarrollar y probar los módulos vamos a necesitar FiveM, el cual podemos descargar de [aquí](#):



FiveM es el que nos va a proporcionar las librerías para poder llevar a cabo el desarrollo.

Pasos a seguir para la ejecución del server

1. Creación de un nuevo directorio en el que descargaremos lo necesario para poder ejecutar el servidor.
Por convención, podemos crear lo siguiente: `./FXServer/server`
2. Descargamos la última build de la rama `master` de [aquí](#). Tener en cuenta que es la versión para Windows.
3. El contenido del `.zip` lo extraemos en la carpeta `server` creada anteriormente.
4. Clonamos [este](#) repositorio en la carpeta `FXServer`, creada anteriormente. Estando en `FXServer`, podemos usar el comando:

```
git clone https://github.com/citizenfx/cfx-server-data.git server-data
```

5. En la carpeta `server-data` creada, creamos un archivo `server.cfg` y lo llenamos con el contenido de ejemplo de [aquí](#). Aún así, en el [anexo](#) de este documento tenemos también dicho contenido.
6. Obtenemos una clave de licencia para poder desarrollar [aquí](#)
7. La añadimos al archivo `server.cfg`, creado anteriormente, junto a la llave `sv_licenseKey` `aqueieberiaIr`
8. Creamos un archivo nuevo, como por ejemplo, `server.bat` y le añadimos lo siguiente:

```
cd C:\Users\nombreUsuario\FXServer\server-data  
C:\Users\nombreUsuario\FXServer\server\FXServer.exe +exec server.cfg
```

Una vez terminados todos los pasos, podemos ejecutar el `server.bat` y debería abrirse una consola, donde veremos que se cargan ya algunos módulos por defecto. En esta consola será desde donde podremos cargar los módulos que vayamos creando. Si todo ha funcionado correctamente, debería mostrar algo parecido a lo siguiente:

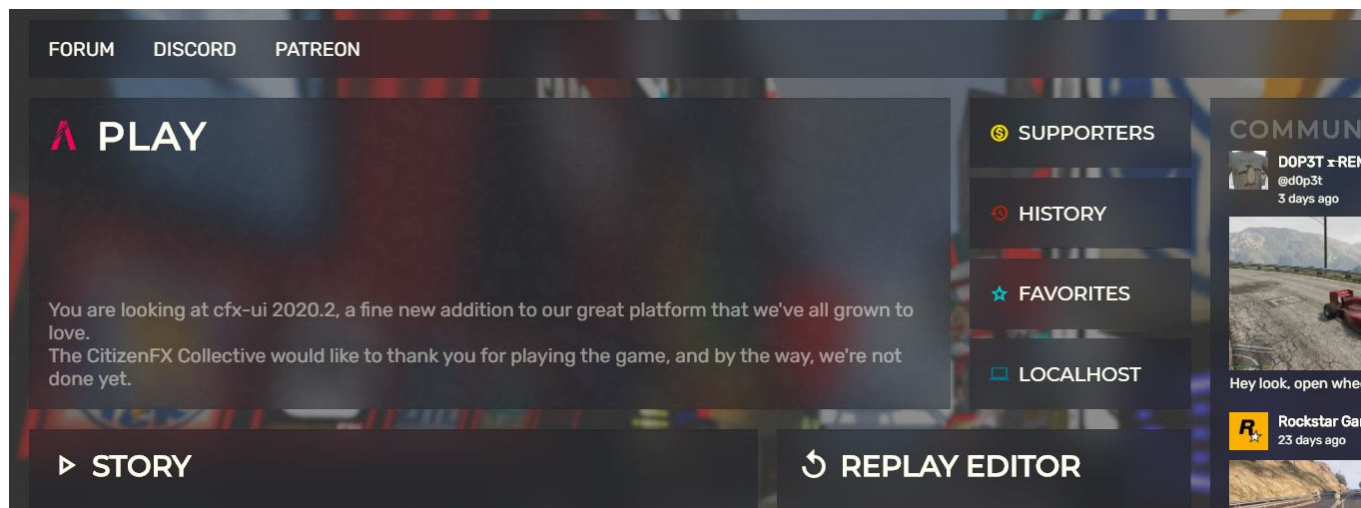
```
Creating script environments for monitor
Started resource monitor
Creating script environments for mapmanager
Started resource mapmanager
Creating script environments for chat
Started resource chat
Started resource spawnmanager
Started gametype Freeroam
Started resource basic-gamemode
Started resource fivem
Creating script environments for hardcap
Started resource hardcap
Creating script environments for rconlog
Started resource rconlog
Started resource scoreboard
Authenticating server license key...
Server license key authentication succeeded. Welcome!
cfx> Sending heartbeat to https://servers-ingress-live.fivem.net/ingress
Started map fivem-map-skater
Started resource fivem-map-skater
Authenticating with Nucleus...
server thread hitch warning: timer interval of 1471 milliseconds
    fff
  cccc ff  xx  xx      rr rr   eee
cc    ffff  xx      rrr  r ee  e
cc    ff    xx  ... rr    eeeee
cccccc ff  xx  xx ... rr    eeeee

Authenticated with cfx.re Nucleus: [REDACTED]
```

Si le damos a `Enter` podremos ver que se puede escribir en la consola. Algunos comandos a recordar son `start nombreModulo`, `stop nombreModulo`, `restart nombreModulo` y `refresh`.

Iniciar FiveM

Ahora ya podemos ejecutar FiveM. Una vez dentro, debemos hacer clic en el botón de localhost, que automáticamente debería conectarse al recién iniciado servidor:



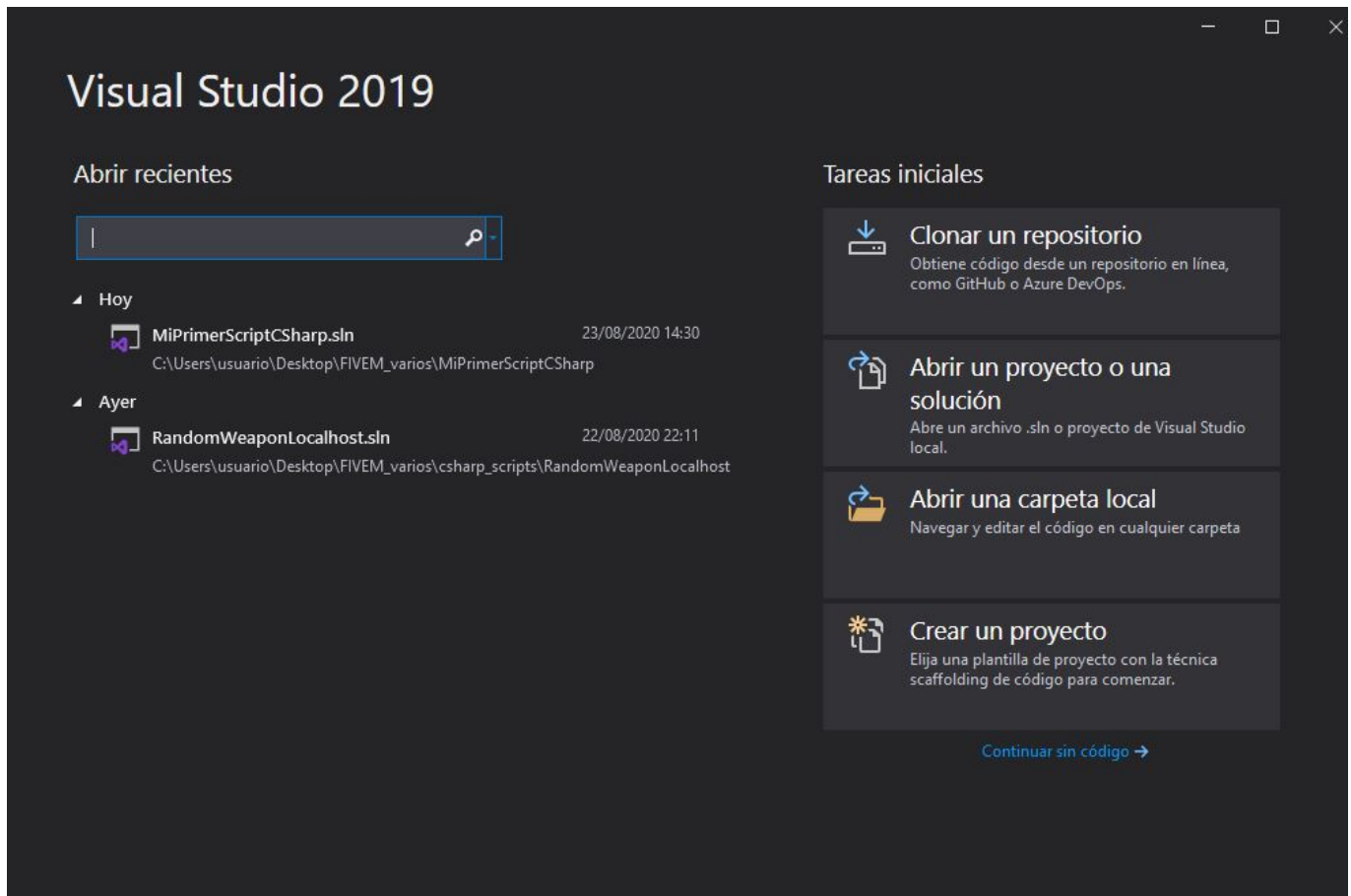
Mientras cargan todos los elementos necesario, veremos una pantalla de carga en la que podremos entretenernos con el movimiento de un monigote mientras esperamos:



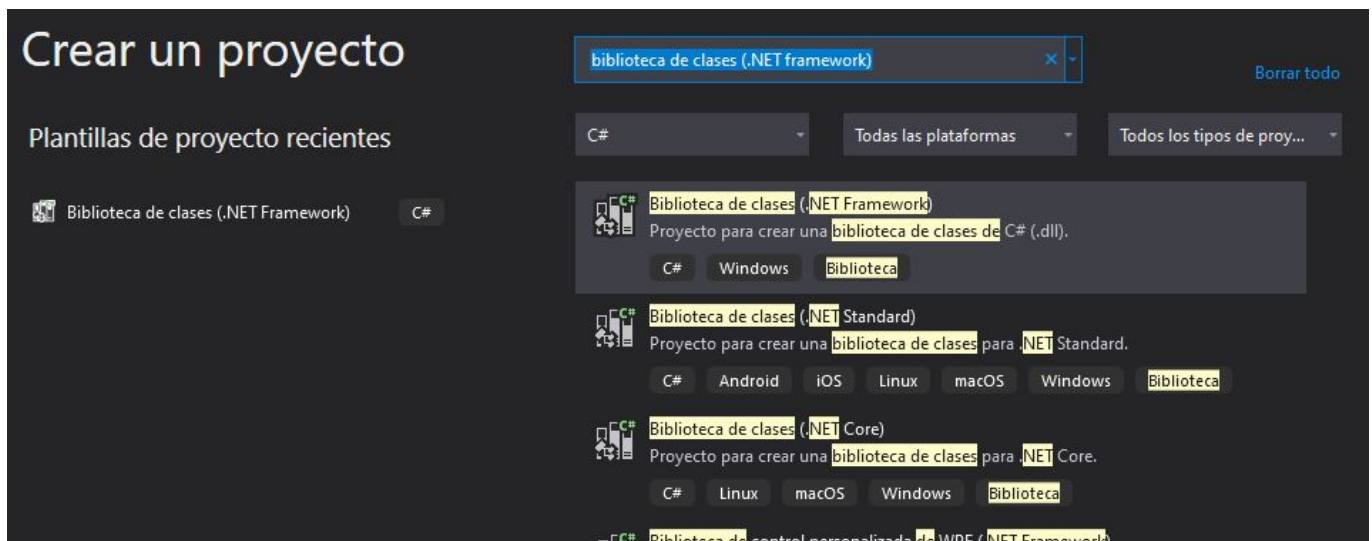
Una vez cargado todo, nos hará aparecer en un lugar aleatorio del mapa. Entonces ya podemos cerciorarnos de que ha funcionado correctamente. Si queremos acceder a la consola dentro del juego usaremos **F8**. Para acceder al chat usaremos la letra **T**.

Creación del primer módulo en C# usando MVSC

Cuando iniciemos el MVSC, lo primero que haremos será crear un nuevo proyecto:



Tras darle a siguiente, buscamos **biblioteca de clases (.NET Framework)** y la escogemos:



En la siguiente pantalla indicamos la configuración que deseamos:

Configure su nuevo proyecto

Biblioteca de clases (.NET Framework) C# Windows Biblioteca

Nombre del proyecto

ClassLibrary1

Ubicación

C:\Users\usuario\source\repos

Nombre de la solución ⓘ

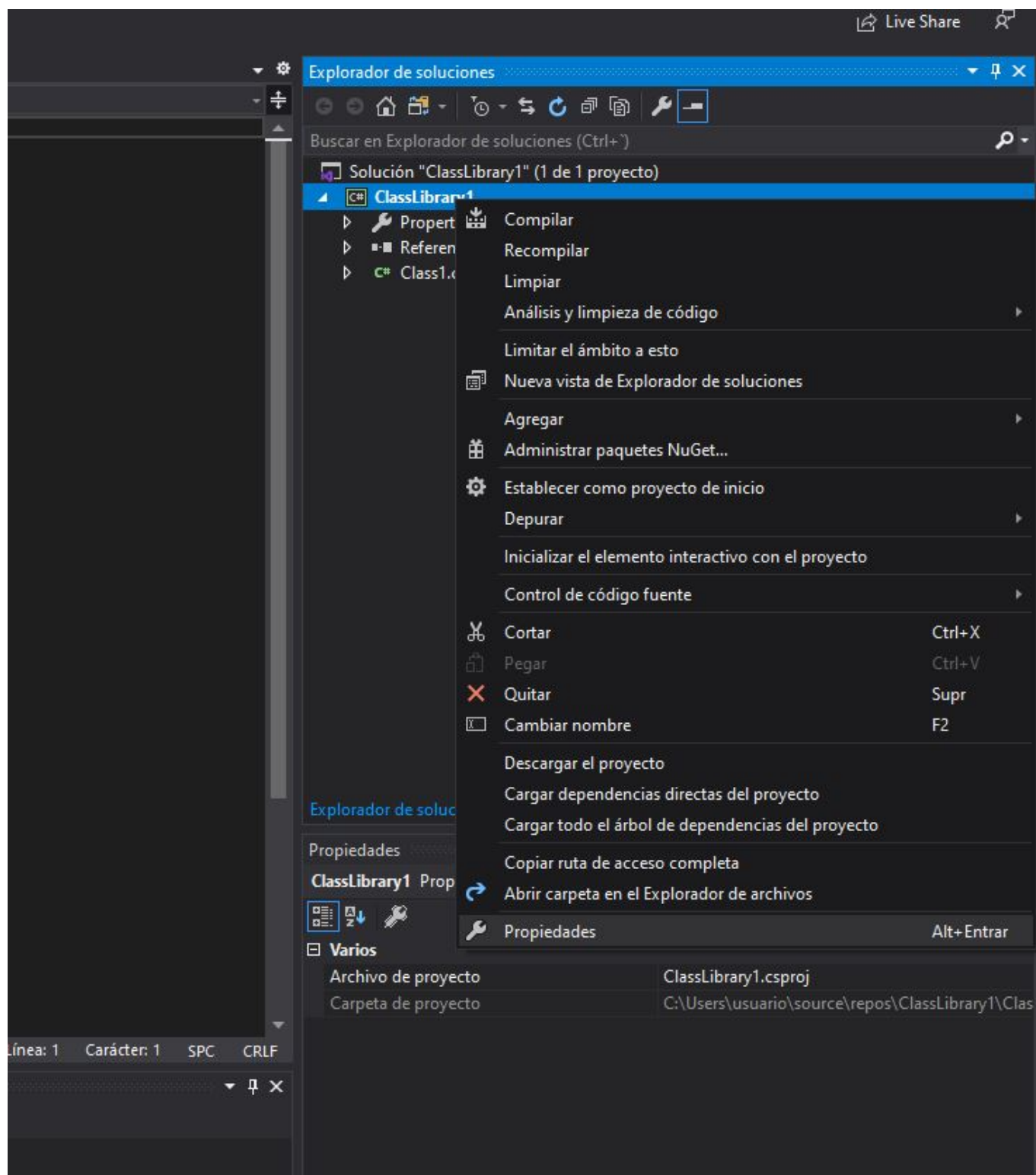
ClassLibrary1

☐ Colocar la solución y el proyecto en el mismo directorio

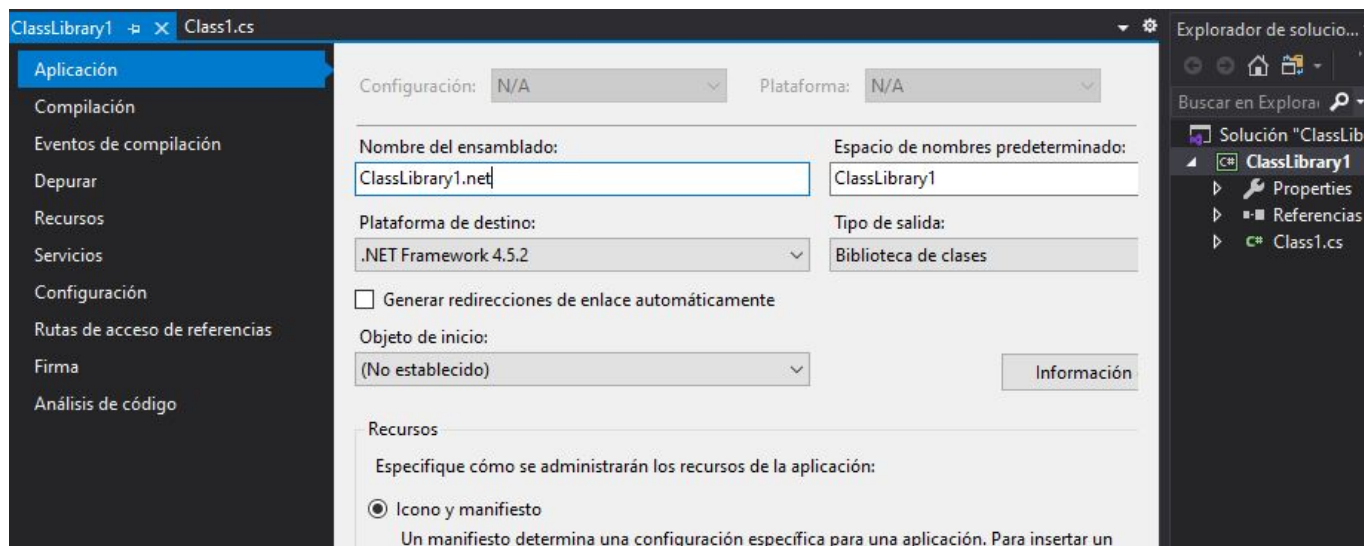
Framework

.NET Framework 4.5.2

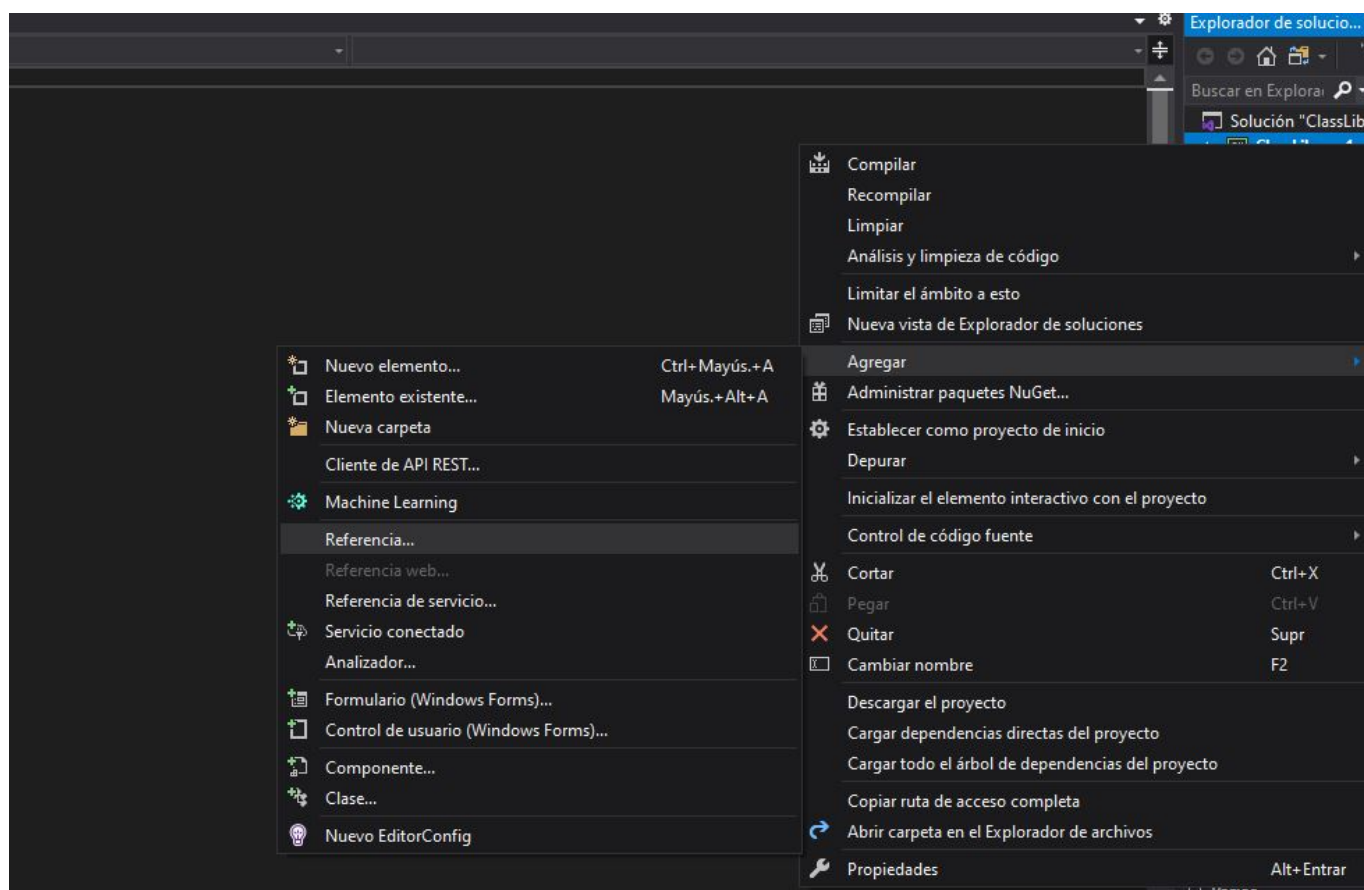
Finalmente, lo creamos. Una vez dentro, hacemos clic con el botón derecho en la clase en el panel derecho, justo debajo de donde pone **Solución nombreClase**, y seleccionamos **Propiedades**:



En el apartado **aplicación** tenemos que añadirle la extensión **.net** al nombre del ensamblado y guardar con **Control+S**:



Por último, volvemos a hacer clic derecho en el mismo lugar pero esta vez vamos a **Agregar** y escogemos **Referencia**:



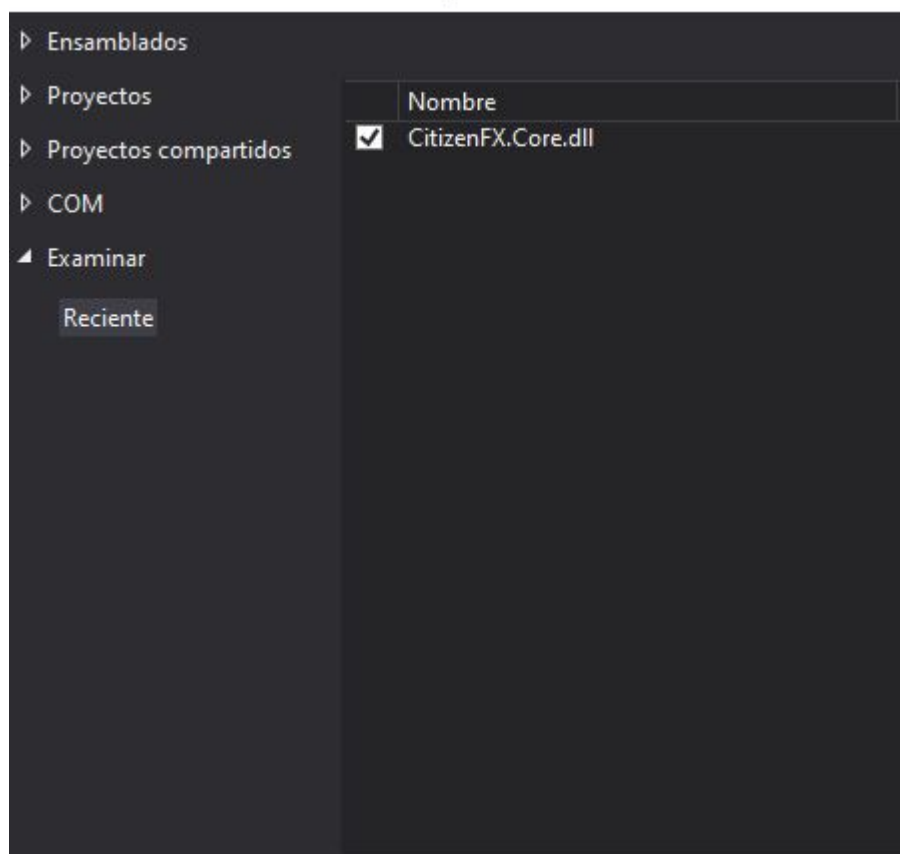
En el apartado **Examinar** tendremos que añadir el **.dll** cuyo nombre es **CitizenFX.Core.dll**. Éste debería encontrarse en la ruta

C:\Users\nombreUsuario\AppData\Local\FiveM\FiveM.app\citizen\clr2\lib\mono\4.5. Lo

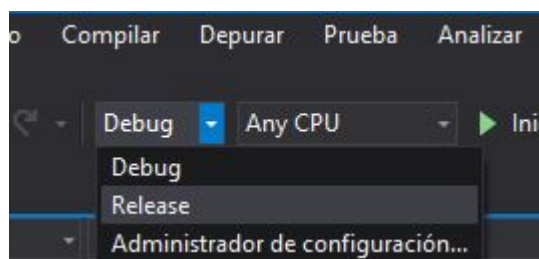
copiamos y lo pegamos en el proyecto de MVSC creado. Tras ello lo añadimos desde el apartado **Examinar**.

Asegurémonos de que está marcado:

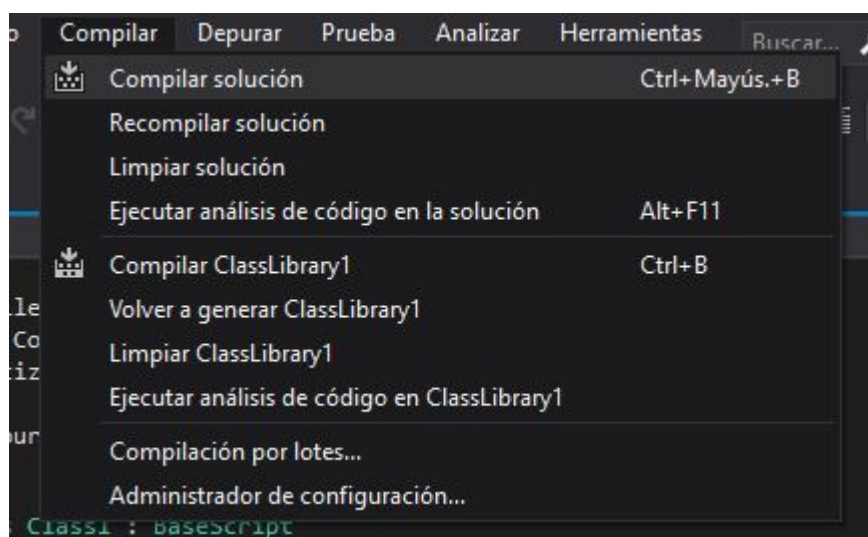
Administrador de referencias: ClassLibrary1



Una vez hecho esto, ya podemos empezar a programar. En el [anexo](#) dejo un código básico de ejemplo, el cual podemos añadir al proyecto. Debería funcionar sin problemas, ya que todas las dependencias están cubiertas. Una vez lo hemos pegado en el archivo creado, seleccionaremos la opción release:



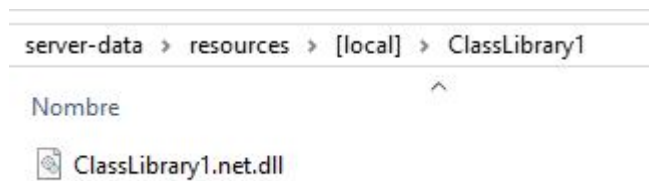
Y compilaremos la solución:



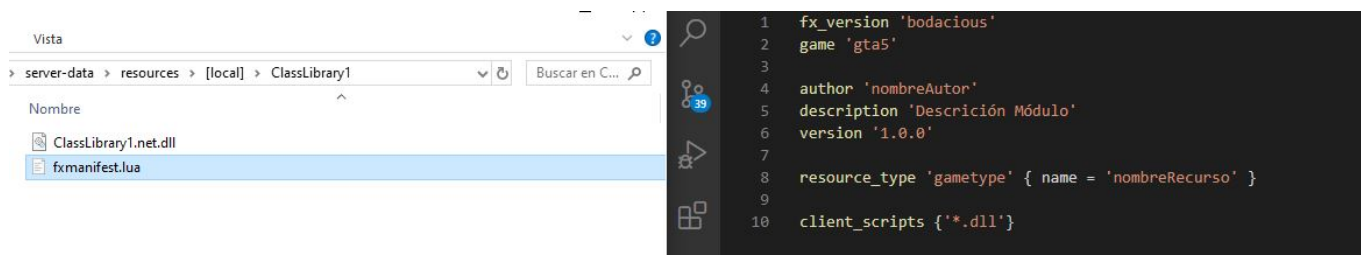
Con esta compilación obtendremos una serie de archivos. El que nos interesa es el que tiene el nombre de nuestra solución, que en mi caso es **ClassLibrary1.net.dll**:

Nombre	Fecha de modificación	Tipo
CitizenFX.Core.dll	20/08/2020 19:07	Extensión de la ap...
CitizenFX.Core.pdb	20/08/2020 19:07	Program Debug D...
CitizenFX.Core.xml	20/08/2020 19:07	Documento XML
ClassLibrary1.net.dll	23/08/2020 17:01	Extensión de la ap...
ClassLibrary1.net.pdb	23/08/2020 17:01	Program Debug D...
MsgPack.dll	29/05/2020 12:54	Extensión de la ap...
MsgPack.xml	29/05/2020 12:54	Documento XML
System.Collections.Immutable.dll	29/05/2020 12:54	Extensión de la ap...
System.Reflection.Metadata.dll	29/05/2020 12:54	Extensión de la ap...

Copiamos dicho **.dll** y lo pegamos en **rutaEscogida\server\server-data\resources\[local]\MiPrimerModulo**, siendo **MiPrimerModulo** una carpeta que creamos, preferiblemente, con el nombre del propio módulo:



Tras esto tenemos que crear un **fxmanifest.lua** con el contenido que indicamos en el [anexo](#). Lo modificamos según nuestras necesidades:



Ahora, desde la consola, hacemos un **refresh** para que detecte el nuevo módulo añadido:

```

cfx> refresh
Found new resource ClassLibrary1 in C:/Users/usuario/Desktop/FIVEM_varios/server/server-data/resources//[local]/ClassLib
rary1
```

Tras ello, iniciamos el módulo:

```

cfx> start ClassLibrary1
Changing gametype from basic-gamemode to ClassLibrary1
Stopping resource fivem-map-skater
Stopping resource fivem
Stopping resource basic-gamemode
Started gametype nombreRecurso
Started resource ClassLibrary1
```


Si ahora volvemos a FiveM, veremos desde la consola de desarrollador que se ha cargado correctamente:

```
script:ClassLibrary1 Loaded ClassLibrary1.net, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null  
script:ClassLibrary1 Instantiated instance of script MyResourceNameClient.Class1.
```

Si vamos al chat del juego y escribimos `/car`:



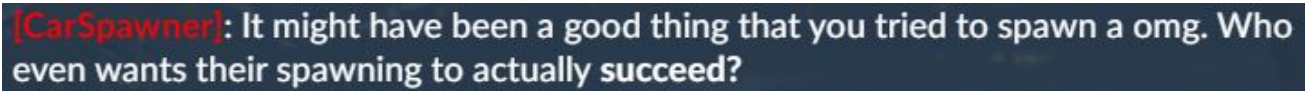
Debería hacer aparecer un coche y nos meterá automáticamente dentro de él. Además de mostrarnos por chat el mensaje que le hemos indicado en el código con el nombre del modelo:



Tener en cuenta que, por defecto, si no le hemos indicado ningún parámetro, nos hace aparecer un coche por defecto, cuyo modelo es el `adder`. Pero si le pasas el modelo deseado como parámetro, lo hará aparecer:



Si, por otro lado, el modelo no existe, entonces lo indica por chat:

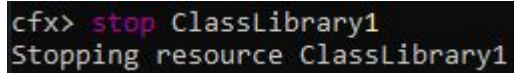


[CarSpawner]: It might have been a good thing that you tried to spawn a omg. Who even wants their spawning to actually succeed?



► /car omg

Si queremos hacer que deje de estar activo el módulo, podemos escribir en la consola `stop nombreModulo`:



```
cfx> stop ClassLibrary1
Stopping resource ClassLibrary1
```

Y vemos que ahora ya no hace nada el comando en FiveM:



Pues con esto se termina el tutorial que debería permitirnos poder desarrollar sin problemas módulos para FiveM.

Anexo

A1

```
# Only change the IP if you're using a server with multiple network interfaces,
otherwise change the port only.
endpoint_add_tcp "0.0.0.0:30120"
endpoint_add_udp "0.0.0.0:30120"

# These resources will start by default.
ensure mapmanager
ensure chat
ensure spawnmanager
ensure sessionmanager
```

```
ensure fivem
ensure hardcap
ensure rconlog
ensure scoreboard

# This allows players to use scripthook-based plugins such as the legacy Lambda
Menu.
# Set this to 1 to allow scripthook. Do note that this does _not_ guarantee
players won't be able to use external plugins.
sv_scriptHookAllowed 0

# Uncomment this and set a password to enable RCON. Make sure to change the
password - it should look like rcon_password "YOURPASSWORD"
#rcon_password ""

# A comma-separated list of tags for your server.
# For example:
# - sets tags "drifting, cars, racing"
# Or:
# - sets tags "roleplay, military, tanks"
sets tags "default"

# A valid locale identifier for your server's primary language.
# For example "en-US", "fr-CA", "nl-NL", "de-DE", "en-GB", "pt-BR"
sets locale "root-AQ"
# please DO replace root-AQ on the line ABOVE with a real language! :)

# Set an optional server info and connecting banner image url.
# Size doesn't matter, any banner sized image will be fine.
#sets banner_detail "https://url.to/image.png"
#sets banner_connecting "https://url.to/image.png"

# Set your server's hostname
sv_hostname "FXServer, but unconfigured"

# Nested configs!
#exec server_internal.cfg

# Loading a server icon (96x96 PNG file)
#load_server_icon myLogo.png

# convars which can be used in scripts
set temp_convar "hey world!"

# Uncomment this line if you do not want your server to be listed in the server
browser.
# Do not edit it if you *do* want your server listed.
#sv_master1 ""

# Add system admins
add_ace group.admin command allow # allow all commands
add_ace group.admin command.quit deny # but don't allow quit
add_principal identifier.fivem:1 group.admin # add the admin to the group
```

```
# Hide player endpoints in external log output.
sv_endpointprivacy true

# enable OneSync with default configuration (required for server-side state
awareness)
onesync_enabled true

# Server player slot limit (must be between 1 and 32, unless using OneSync)
sv_maxclients 32

# Steam Web API key, if you want to use Steam authentication
(https://steamcommunity.com/dev/apikey)
# -> replace "" with the key
set steam_webApiKey ""

# License key for your server (https://keymaster.fivem.net)
sv_licenseKey changeme
```

A2

```
using System;
using System.Collections.Generic;
using CitizenFX.Core;
using static CitizenFX.Core.Native.API;

namespace MyResourceNameClient
{
    public class Class1 : BaseScript
    {
        public Class1()
        {
            EventHandlers["onClientResourceStart"] += new Action<string>
(OnClientResourceStart);
        }

        private void OnClientResourceStart(string resourceName)
        {
            if (GetCurrentResourceName() != resourceName) return;

            RegisterCommand("car", new Action<int, List<object>, string>(async
(source, args, raw) =>
            {
                // account for the argument not being passed
                var model = "adder";
                if (args.Count > 0)
                {
                    model = args[0].ToString();
                }

                // check if the model actually exists
                // assumes the directive `using static CitizenFX.Core.Native.API;`
            }
            )
            );
        }
    }
}
```

```

var hash = (uint)GetHashKey(model);
if (!IsModelInCdimage(hash) || !IsModelAVehicle(hash))
{
    TriggerEvent("chat:addMessage", new
    {
        color = new[] { 255, 0, 0 },
        args = new[] { "[CarSpawner]", $"It might have been a good
thing that you tried to spawn a {model}. Who even wants their spawning to actually
^*succeed?" }
    });
    return;
}

// create the vehicle
var vehicle = await World.CreateVehicle(model,
Game.PlayerPed.Position, Game.PlayerPed Heading);

// set the player ped into the vehicle and driver seat
Game.PlayerPed.SetIntoVehicle(vehicle, VehicleSeat.Driver);

// tell the player
TriggerEvent("chat:addMessage", new
{
    color = new[] { 255, 0, 0 },
    args = new[] { "[CarSpawner]", $"Woohoo! Enjoy your new ^*
{model}!" }
});
}), false);
}
}
}

```

A3

```

fx_version 'bodacious'
game 'gta5'

author 'nombreAutor'
description 'Descripción Módulo'
version '1.0.0'

resource_type 'gametype' { name = 'nombreRecurso' }

client_scripts {'*.dll'}

```