# Intro

En la Build puede haber problemas. Por ejemplo

* Referencia a Assets que no están (al editor le puede dar igual pero a la Build no)

BlueprintUtilities: clase en C++ que expone llamadas nuevas a BP, que no están referenciadas a una clase de C++.

# Build

Vamos a hacer una Build desde el Editor y luego SIN.

Table

Description automatically generated

Estas 3 carpetas son las que se puede borrar.

* Intermediate: en Build se guarda los directorios con los obj
  + Con los Includes y mierdas
* Saved
* DerivedDataCache

**Project Launcher**: para compilar o lanzar el proyecto.

A screenshot of a computer

Description automatically generated

Tipo de BUILD

Graphical user interface, application, Teams

Description automatically generated

¿Shipping?

**Cook**

How to cook

tengo assets en el editor. Unreal se lo prepara para que en el editor se vea. Se generan mipmaps, texturas, etc (se cocinan) y se genera una versión de la textura especial para la Play4, o para IOS, por ejemplo. Y se descartan los datos del editor.

* **By the book**: todo, se borra todo lo cocinado y volvemos a empezar
* **On the fly**: cuando tienes una Play4 . Tengo los datos en el PC cocinados, y le digo a la play que lo busque en local, si no al servidor de cooking 🡪 VA BIEN si los cambios son incrementales. Solo se baja lo que se ha cambiado. Es mas lento arrancar una build que no está paquetizada. ¿?
* **Do not cook**: lo que haya es lo que está, se usa cuando solo toco C++. Si solo toco C++ no tengo que cocinar nada.

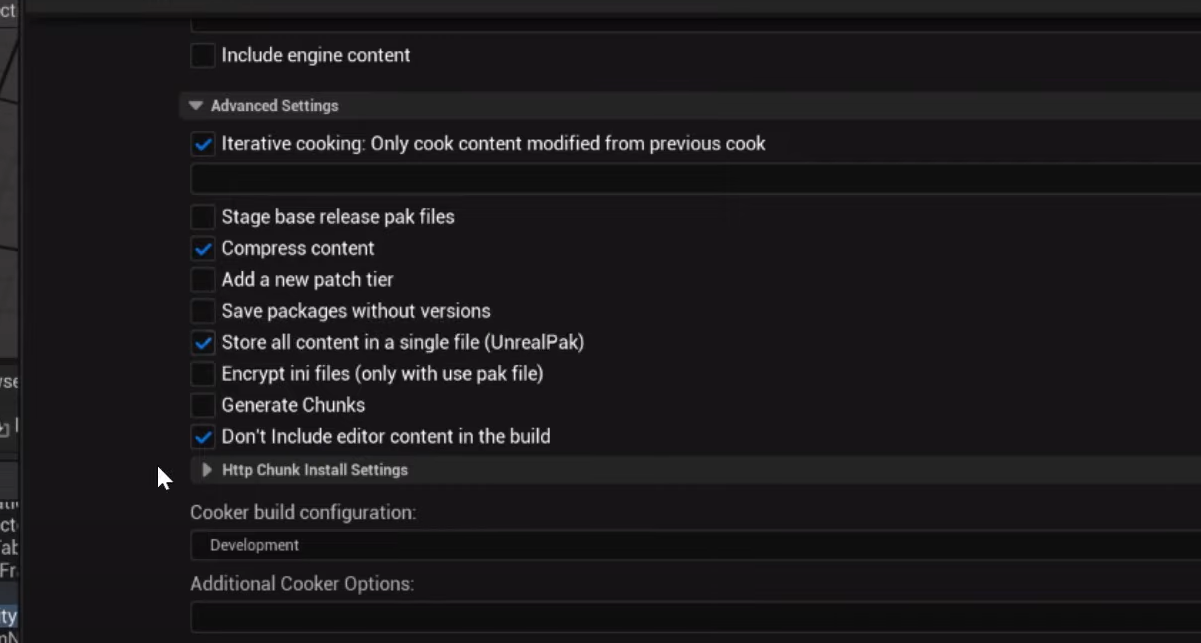
Cooked platform

* Windows: la marcamos
* Windows Editor build para el editor
* Windows Server: datos que necesita un servidor (las texturas no las necesita)
* Windows Client: todo menos lo que necesita el Server

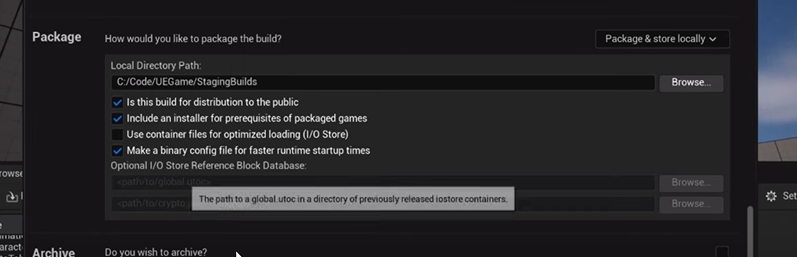
Cooked Maps: si no lo sé, le ponemos todos

Iterative cooking: ACTIVAR, solo cocinas lo que ha modificado

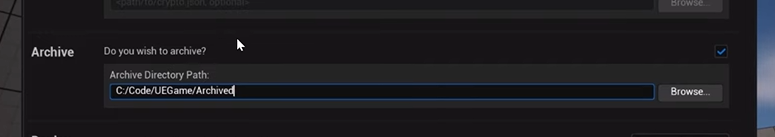
Dont include editor content in the build: ACTIVAR



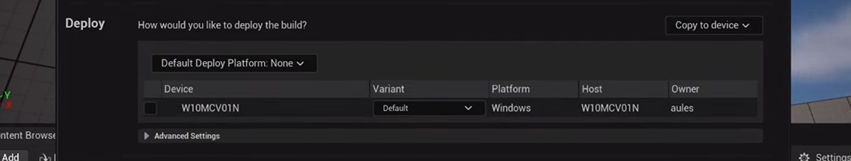
Package



Archive:



Deploy: para instalarlo en algún sitio (de momento no hará falta)



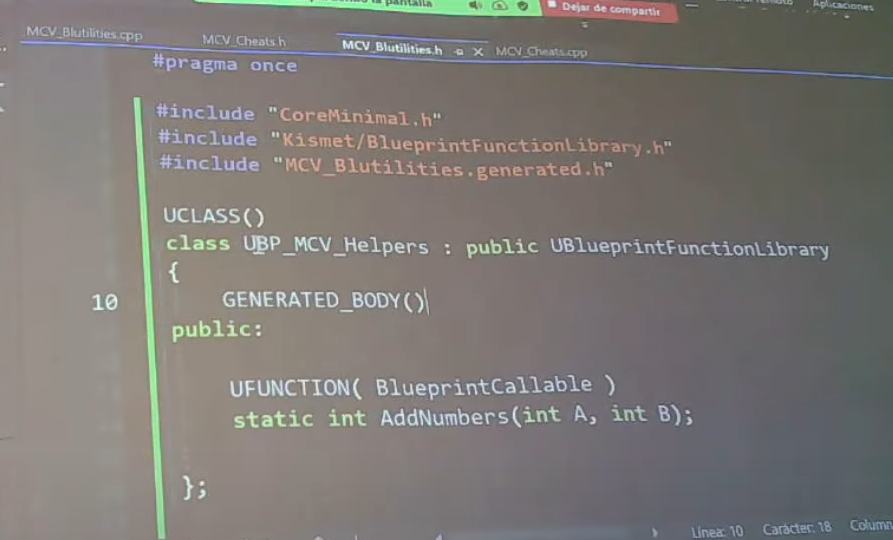
Launch: no hace falta que lo lances

Text

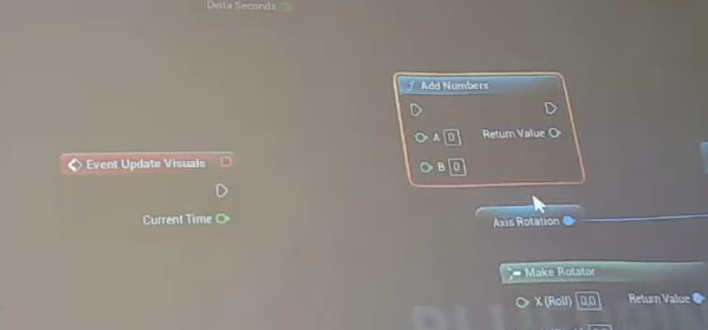
Description automatically generated

El ini se actualiza. Lo ha tocado en Project setting o algo así.

# Blueprint Utilities



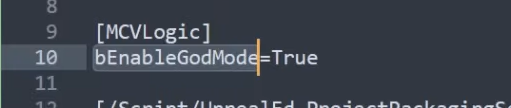
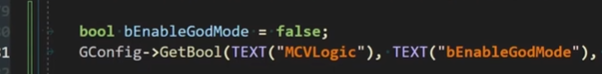
Ahora ya Podemos llamarla desde Blueprint

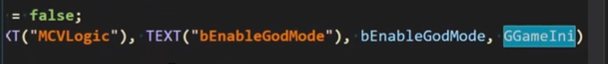


No se puede mandar ptr a structs. CREO que deben ser UOBJECT

# Leemos un ini

En el Default.ini





Está bien porq en una build cerrada puedo tocar cosas cambiando el .ini

En el Shipping estás cosas se desactivan!!!

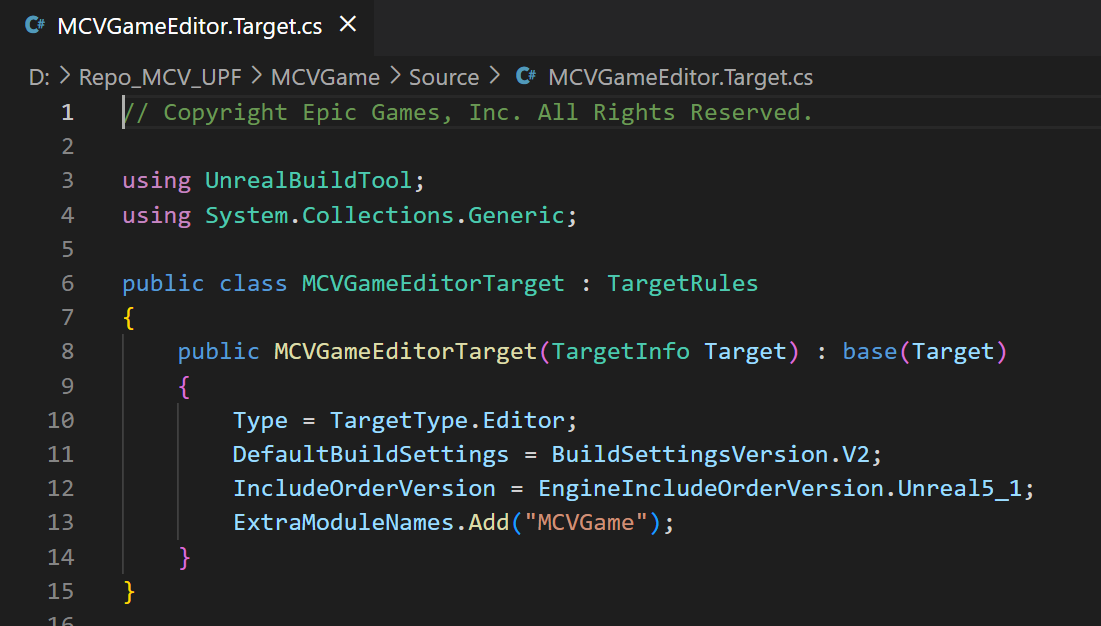
# MODULOS Y PLUGINS

El editor contiene al juego. El juego no contiene al editor.

Es bueno separar las cosas

Se pone en: UEGame/MCVGame/Source

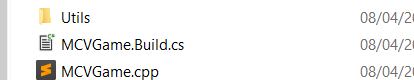
Hay un MCVGameEditorTarget



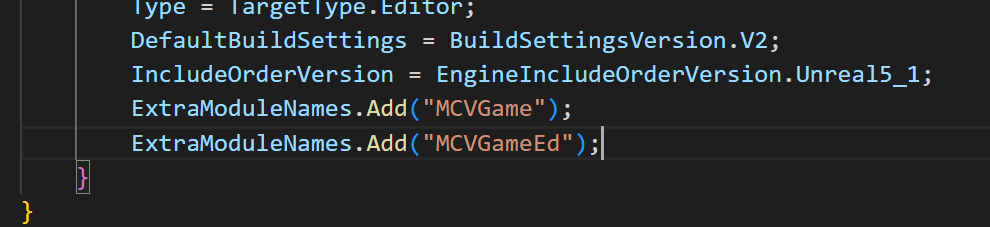
Pide el modulo MCVGame (línea 13). Nosotros montamos ahora otro modulo y le decimos en la línea 13 que tamb lo requiera.

**Crea carpeta MCVGameEd**

Se va a MCVGame/Source/MCVGame y **copia el MCVGameBuild** en la carpeta



Y ahora añadimos el otro



Un plugin proporciona módulos y un juego tiene un modulo que requiere otros módulos

El editor es un target más que requiere otros módulos.

El editor usa el juego y el juego usa módulos de plugins

El editor puede usar todo lo que usa el juego.

Explicación:

* JUEGO

[Target Game] requieren modulos. (Modulo Game + Plugins (con sus módulos tamb)).

* EDITOR (es un ejecutable que busca targets. Y dice que necesita 2 modulos )

El Engine Editor usa el juego entero.

* + Hay un target GameEditor (que contiene el juego entero)
  + Y luego estan tamb los módulos
    - My GameEd Module
  + Y hay plugins
    - Plugin Editor con los módulos Ed

Importante: Tenemos cosas en el editor que se quedan en el editor (y no me interesa que estén en el juego). Por ejemplo un Actor que cree arboles. Yo lo quiero para el editor. Luego en el juego deben estar todos ya creados.