



Multimedia

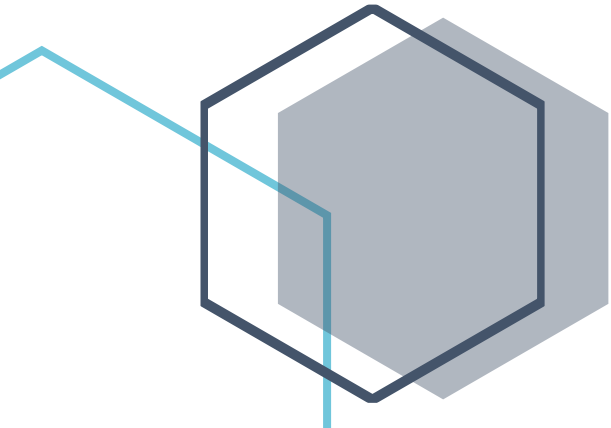
Minecraft interactivo

Documentación
**Desarrollo del Software
de Laboratorio**

ALBERTO GARCÍA APARICIO

JOSÉ LUIS CABEZAS VELASCO

Curso 2022/2023



Contenido

1. Descripción general.....	2
2. Requisitos de aplicación	2
3. Decisiones y justificación de diseño e implementación.....	2
3.1. Herramientas utilizadas	2
3.2. Diseño.....	3
3.3 Implementación	6
4. Conclusiones.....	9
Bibliografía	9
Manual de usuario (guía de inicio).....	9

1. Descripción general

Minecraft interactivo es una aplicación multimedia de escritorio, en la cual mediante los diferentes elementos como pueden ser preguntas de rellenar, encuestas y demás podremos vivir la aventura de nuestro personaje a través del mundo de Minecraft.

En este juego nuestras acciones tendrán consecuencias, a tal punto que nos podrá llevar a la muerte, obligándonos a volver a reiniciar nuestra aventura, el objetivo es sobrevivir hasta el final de esta pequeña historia contestando correctamente a las diferentes preguntas.

2. Requisitos de aplicación

Los requisitos para el arranque de esta aplicación son:

- Tener instalado Visual Studio 2019, y arrancar el programa con Debug|x86.
- 36 KB para la instalación.
- Windows 10 (y posteriores)

3. Decisiones y justificación de diseño e implementación

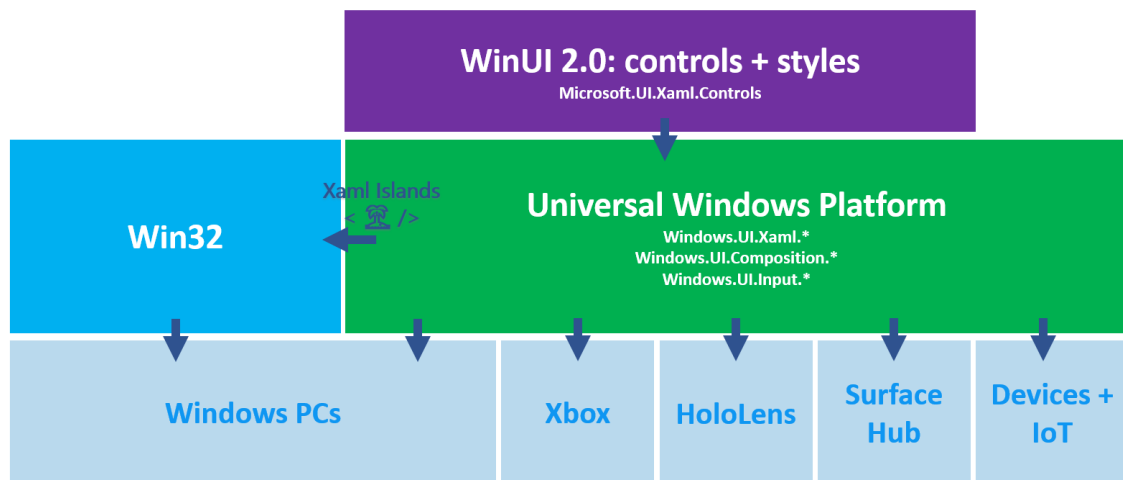
3.1. Herramientas utilizadas

Para realizar esta aplicación hemos optado por utilizar la tecnología UWP, las API principales de la UWP son las mismas en todos los dispositivos de Windows de forma que una aplicación UWP se ejecutará en cualquier dispositivo de Windows 10, independientemente de si es un ordenador, una tablet, una Xbox, unas gafas de realidad virtual, etc. [1]

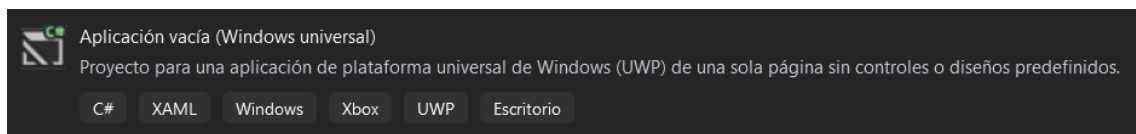


Esta tecnología aporta una capacidad adaptativa a los controles, permitiendo que estos se adapten fácilmente a cualquier pantalla, ya que esta ejecuta la aplicación en términos de escalado y también de diseño.

También utilizaremos una librería de Windows, WinUI2, que provee algunos controles y elementos de interfaz para aplicaciones UWP. [2]



Estas herramientas las utilizaremos mediante el IDE Visual Studio 2019, en el cual crearemos un proyecto Windows Universal, UWP, en el cual se administran los elementos del Front-end mediante XAML y del Back-end mediante C#.

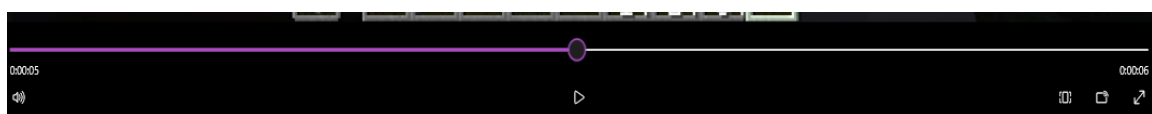


El principal motivo por el que usaremos estas herramientas es por que anteriormente ya hemos realizado otros proyectos académicos, y los que nos habían resultado más cercanos al usuario eran las aplicaciones de escritorio mediante WPF o UWP, y de estas dos hemos elegido UWP debido a la adaptabilidad que tiene, y por la librería WinUI2 puesto que en esta podemos encontrar cosas como los elementos para el control del video, lo cual ayuda bastante al avance del trabajo y da una imagen mas profesional de la aplicación.

3.2. Diseño

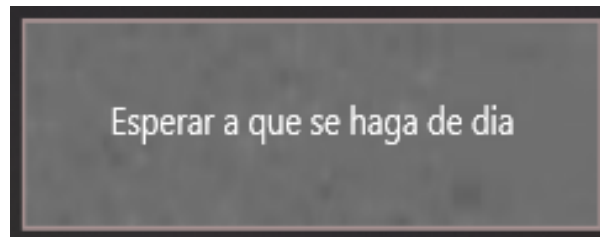
De primeras encontramos el video en reproducción, en la aplicación podemos encontrar los principales controles que estarán a lo largo de la aplicación:

- Ajustador de volumen
- Botón de Play/Stop
- Botón de Relación de aspecto
- Botón de Transmitir en dispositivo
- Botón de Pantalla completa
- Slider para la reproducción del video
- Tiempo reproducido y tiempo restante

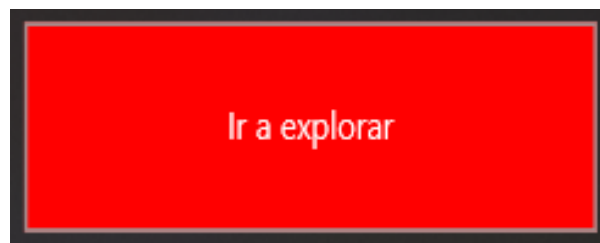


Hemos elegido estos debido a que creemos que son los necesarios, y después hemos añadido el de relación de aspecto y el de transmisión para tener unas funcionalidades extras que permitan darle un aspecto mas profesional, ya que nos permite eliminar las bandas negras y dejar toda la aplicación rellena con el video, y poder transmitir el video en otro dispositivo.

Encontraremos también a lo largo de la aplicación botones con las mismas características que los que encontramos en Minecraft, para que sea lo más próximo al juego. **[3]**



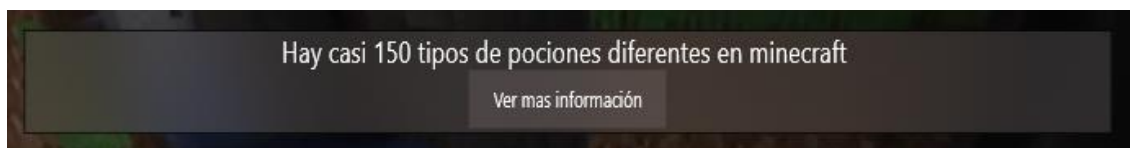
Y el cual después de seleccionar una decisión incorrecta se volverá rojo.



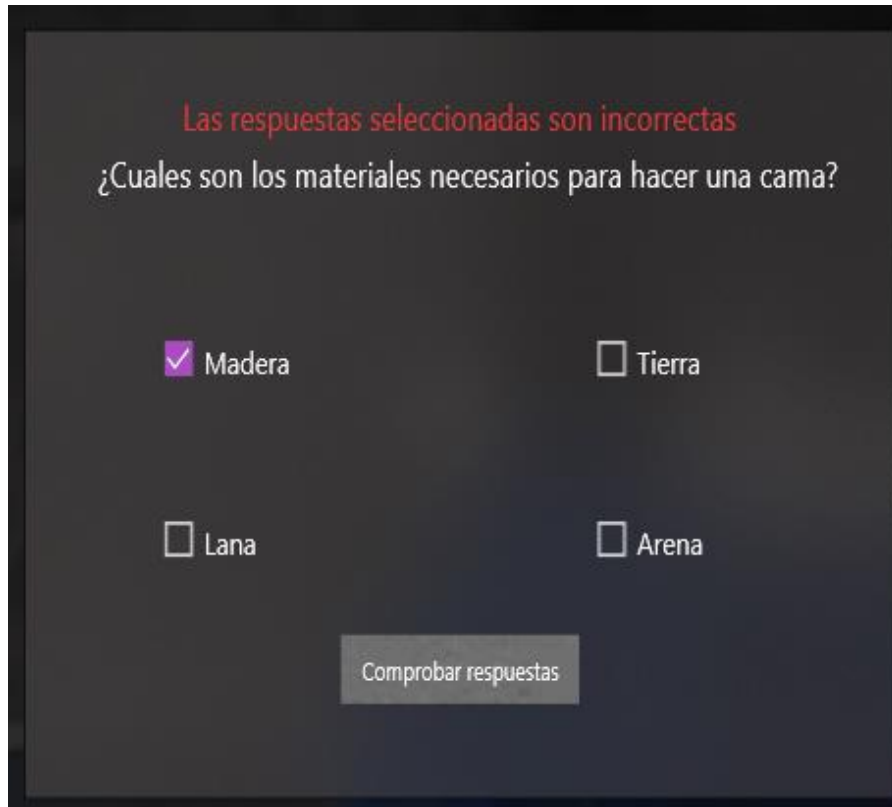
Encontráremos juegos tipo ahorcado para pasar al siguiente nivel, en el cual a cada fallo que se produzca se irán dando pistas al jugador.



También existen algunos comentarios sobre curiosidades de Minecraft situados en la parte superior de la aplicación, el cual tendrán un enlace que abrirá una nueva pestaña en el navegador predeterminado del equipo del jugador para obtener más información. **[4]**



También hay preguntas de selección múltiple, en la cual al comprobar las respuestas si estas son correctas saldrá un dialogo informándote de que has cometido un error.



Las respuestas seleccionadas son incorrectas

¿Cuales son los materiales necesarios para hacer una cama?

☒ Madera

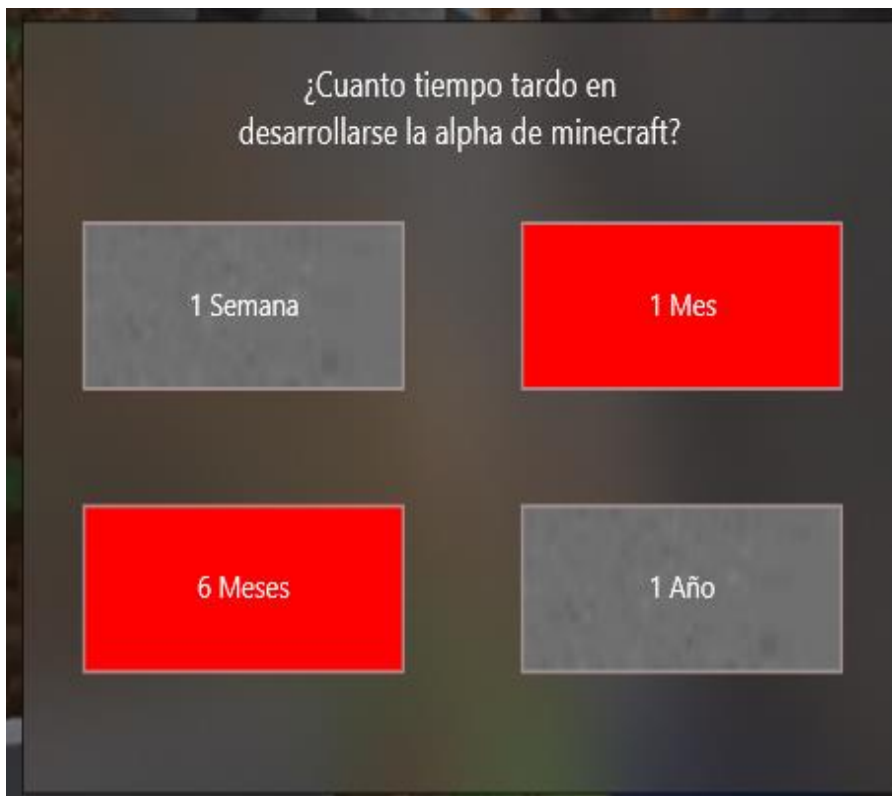
☐ Tierra

☐ Lana

☐ Arena

Comprobar respuestas

Otro de los elementos son las preguntas con múltiples respuestas, las cuales se irán marcando en rojo como hemos mencionado anteriormente en caso de error.



¿Cuanto tiempo tardo en desarrollarse la alpha de minecraft?

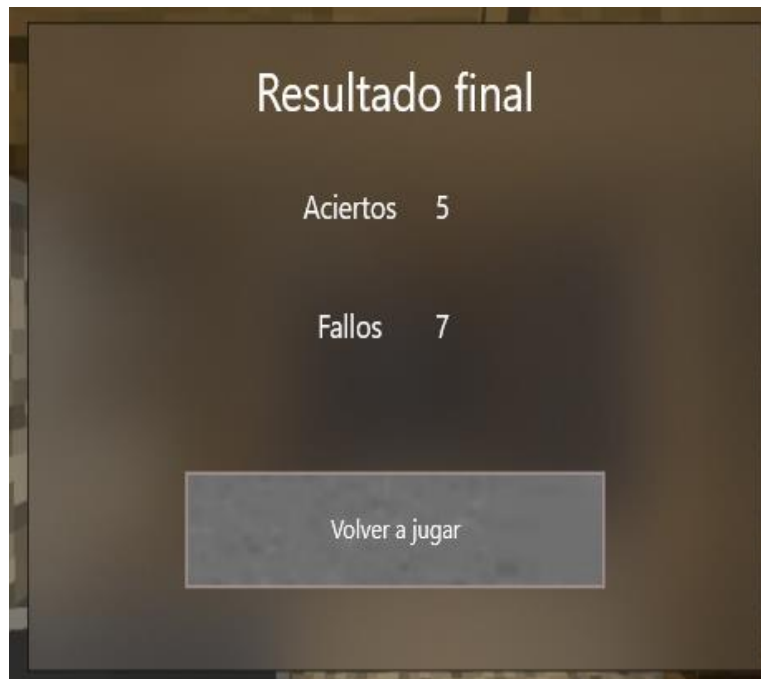
1 Semana

1 Mes

6 Meses

1 Año

Por último, nos encontramos la pantalla final en la cual encontraremos los aciertos y los fallos que el jugador ha tenido a lo largo de la aplicación, y una opción para volver a jugar.



3.3 Implementación

En la implementación primero encontramos unas variables globales para manejar los fallos y aciertos generales que se mostraran al final del juego, una para manejar los fallos del ahorcado para ir dando pistas, otra variable del acto en la que nos encontramos, y por último el color rojo para pintar los botones cuando se selecciona la respuesta incorrecta.

```
int fallos = 0;
int fallosTarea2 = 0;
int aciertos = 0;
int acto = 1;
SolidColorBrush redBrush = new SolidColorBrush(Windows.UI.Colors.Red);
```

Los actos se seleccionan mediante un switch el cual activara las diferentes tareas correspondientes, hemos elegido esta forma puesto que en muchas tareas se necesita volver a un acto anterior y la forma mas facil era dejandolos todos en un switch para ir accediendo a estos mas facilmente.

```
private void video_MediaEnded(object sender, RoutedEventArgs e)
{
    switch (acto)
    {
        case 1:
            tarea1();
            break;
        case 2:
            tarea2();
            break;
        case 100:
            Boton_Reintentar1.Visibility = Visibility.Visible;
            break;
        case 3:
            tarea3();
            break;
        case 4:
            tarea4();
            break;
        case 5:
            tarea5();
            break;
        case 500:
            Boton_Reintentar6.Visibility = Visibility.Visible;
            break;
        case 6:
            tarea6();
            break;
        default:
            // code block
            break;
    }
}
```

En la mayoría de las tareas lo principal es quitar y dar visibilidad a los diferentes elementos de control que encontramos en la aplicación, ir anotando los aciertos y fallos de estos, y seleccionar el acto en el que nos encontramos.

```
private void Boton_Dialog1_2_Click(object sender, RoutedEventArgs e)
{
    fallos++;
    acto = 100;
    video.Play();
    selector_de_video(2);
    Boton_Dialog1_1.Visibility = Visibility.Collapsed;
    Boton_Dialog1_2.Visibility = Visibility.Collapsed;
    Boton_Dialog1_2.Background = redBrush;
    Rectangulo_Dialog.Visibility = Visibility.Collapsed;
}
```

Las únicas que varían mas serían la implementación del ahorcado y de la respuesta múltiple, en la que en el ahorcado encontramos un switch para ir proporcionando información a medida que el usuario falle, y en el de respuesta múltiple se comprueba que los únicos seleccionados son las respuestas correctas.

```
if (TextBox_Dialog3.Text.ToLower() == "bruja")
{
    aciertos++;
    acto = 3;
    video.Play();
    selector_de_video(3);
    Rectangulo_Dialog.Visibility = Visibility.Collapsed;
    Imagen_Dialog3.Visibility = Visibility.Collapsed;
    TextBox_Dialog3.Visibility = Visibility.Collapsed;
    error_Dialog3.Visibility = Visibility.Collapsed;
    Text_Dialog3.Visibility = Visibility.Collapsed;
    Boton_Dialog3.Visibility = Visibility.Collapsed;
    botonEnlace.Visibility = Visibility.Visible;
    Text_Link.Visibility = Visibility.Visible;
    Rectangulo_Link.Visibility = Visibility.Visible;
}
else
{
    fallos++;
    fallosTarea2++;
    switch (fallosTarea2)
    {
        case 1:
            error_Dialog3.Visibility = Visibility.Visible;
            break;
        case 2:
            error_Dialog3.Text = "Respuesta incorrecta: B _ _ j _";
            break;
        case 3:
            error_Dialog3.Text = "Respuesta incorrecta: B _ u j _";
            break;
        case 4:
            error_Dialog3.Text = "Respuesta incorrecta: Bruj _";
            break;
        default:
            // code block
            break;
    }
}
```

```
private void Boton_Dialog4_Click(object sender, RoutedEventArgs e)
{
    if (CheckBox1_Dialog4.IsChecked == true && CheckBox3_Dialog4.IsChecked == true
        && CheckBox2_Dialog4.IsChecked == false && CheckBox4_Dialog4.IsChecked == false)
    {
        // code block
    }
}
```


En el ultimo acto antes de la pantalla final encontramos dos videos que se irán intercambiando entre sí en el mismo error para que no sea todo tan linear, y que cambie dependiendo de los fallos que tiene el jugador.

```
private void Boton_Dialog6_1_Click(object sender, RoutedEventArgs e)
{
    fallos++;
    acto = 500;
    video.Play();
    if (fallos % 2 == 0)
        selector_de_video(51);
    if (fallos % 2 != 0)
        selector_de_video(52);
    Rectangulo_Dialog.Visibility = Visibility.Collapsed;
    Boton_Dialog6_1.Visibility = Visibility.Collapsed;
    Boton_Dialog6_2.Visibility = Visibility.Collapsed;
}
```

En el acto final encontramos el botón de reiniciar que pone todo como al principio.

```
private void Boton_Reiniciar_Click(object sender, RoutedEventArgs e)
{
    Rectangulo_Dialog.Visibility = Visibility.Collapsed;
    Text_Dialog7.Visibility = Visibility.Collapsed;
    Text_Dialog7_Aciertos.Visibility = Visibility.Collapsed;
    Text_Dialog7_AciertosCantidad.Visibility = Visibility.Collapsed;
    Text_Dialog7_Fallos.Visibility = Visibility.Collapsed;
    Text_Dialog7_FallosCantidad.Visibility = Visibility.Collapsed;
    Boton_Reiniciar.Visibility = Visibility.Collapsed;
    acto = 1;
    fallos = 0;
    aciertos = 0;
    video.Play();
    selector_de_video(0);
}
```

Y los métodos auxiliares que sirven para abrir enlaces externos y cargar un nuevo video en el reproductor multimedia.

```
private void botonEnlace_Click(object sender, RoutedEventArgs e)
{
    string enlace = "";
    switch (acto)
    {
        case 3:
            // Create a Uri object from a URI string
            enlace = @"https://minecraft.fandom.com/wiki/Brewing";
            break;
        default:
            // code block
            break;
    }
    var uri = new Uri(enlace);
    abrirEnlace(uri);
}

async void abrirEnlace(Uri uri)
{
    // Launch the URI
    var success = await Windows.System.Launcher.LaunchUriAsync(uri);
}
```

```
public void selector_de_video(int num_video)
{
    string pathAux = "";
    string path = System.AppDomain.CurrentDomain.BaseDirectory.ToString();
    path = path.Replace(@"\bin\Debug\", "");

    switch (num_video)
    {
        case 0:
            pathAux = path + "/Assets/Videos/minecraft_1.mp4";
            break;

        video.Source = new System.Uri(pathAux.ToString());
    }
}
```

4. Conclusiones

La aplicación tiene gran cantidad de los elementos que con UWP podemos integrar, se ha quedado una aplicación con una gran capacidad de adaptabilidad y muy vistosa para que cualquier persona pueda utilizarlo sin requerir una gran habilidad ya que la complejidad del programa es muy pequeña.

Hemos encontrado algunas dificultades principalmente a la hora de organizar los actos, puesto que al ser seleccionados mediante un switch mediante int no podían cobrar un significado específico.

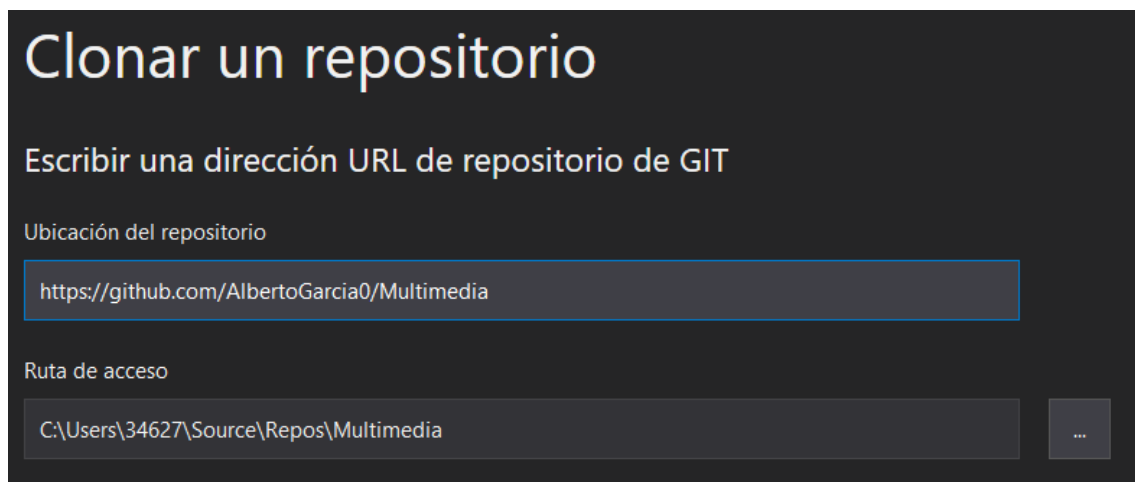
Pensamos que UWP también nos ha limitado a la hora de utilizar algunas características, puesto que el arrastrar elementos por mucho que hemos buscado información en internet no había nada funcional que nos permitiera desarrollar un poco y poder tener algunos elementos adicionales que probablemente con una página web si podríamos haber desarrollado.

Bibliografía

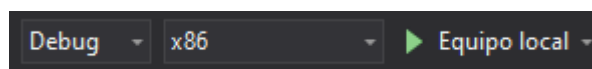
1. <https://learn.microsoft.com/en-us/windows/uwp/>
2. <https://learn.microsoft.com/en-us/windows/apps/winui/winui2/>
3. <https://www.minecraft.net/es-es/about-minecraft>
4. <https://www.hobbyconsolas.com/reportajes/curiosidades-sorprendentes-minecraft-ni-fanaticos-conocen-932909>

Manual de usuario (guía de inicio)

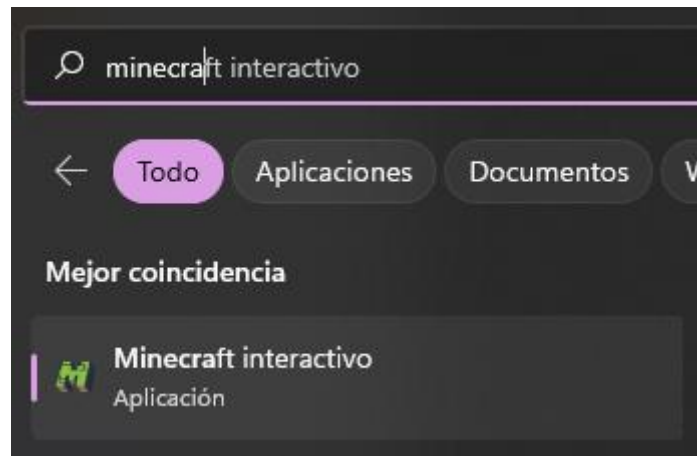
Para ejecutar el proyecto debemos de clonar el repositorio **[Enlace]** desde Visual Studio 2019



Y se pondrán las opciones de Debug y x86 y se ejecutara en el equipo local.



Una vez realizado esto se instalará el programa en nuestro ordenador y ya podremos acceder a él desde el navegador de Windows.



Una vez iniciada la aplicación simplemente deberemos interactuar con los diferentes elementos que encontramos en el programa.

