

**IES AUGUSTO GONZALEZ DE LINARES**

**DEPARTAMENTO DE INFORMATICA**

INVESTIGACIÓN SOBRE DESARROLLO MOVIL

**PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES**

**GRADO SUPERIOR DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

2023/2024

**Díez de Paulino, Albano**

Índice

[1. **RECURSO DE APRENDIZAJE 1 (RA1)** 2](#_Toc147533507)

[1.1. **(CE-C)** Se han identificado las tecnologías de desarrollo de aplicaciones para dispositivos móviles. 2](#_Toc147533508)

[1.2. **(CE-D)** Se han instalado, configurado y utilizado entornos de trabajo para el desarrollo de aplicaciones para dispositivos móviles. 3](#_Toc147533509)

[1.3. **(CE-E)** Se han identificado configuraciones que clasifican los dispositivos móviles en base a sus características. 4](#_Toc147533510)

[1.4. **(CE-G)** Se ha analizado la estructura de aplicaciones existentes para dispositivos móviles identificando las clases utilizadas. 4](#_Toc147533511)

[1.5. **(CE-H)** Se han realizado modificaciones sobre aplicaciones existentes. 5](#_Toc147533512)

[1.6. **(CE-I)** Se han utilizado emuladores para comprobar el funcionamiento de las aplicaciones. 6](#_Toc147533513)

[2. **RECURSO DE APRENDIZAJE 2 (RA2)** 8](#_Toc147533514)

[2.1. **(CE-A)** Se ha generado la estructura de clases necesaria para la aplicación. 8](#_Toc147533515)

[2.2.**(CE-B)** Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas. 9](#_Toc147533516)

[2.3. **(CE-C)** Se han utilizado las clases necesarias para la conexión y comunicación con dispositivos inalámbricos. 11](#_Toc147533517)

[2.4.**(CE-D)** Se han utilizado las clases necesarias para el intercambio de mensajes de texto y multimedia. 11](#_Toc147533518)

[2.5. **(CE-E)** Se han utilizado las clases necesarias para establecer conexiones y comunicaciones HTTP y HTTPS. 12](#_Toc147533519)

[2.6. **(CE-F)** Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia. 13](#_Toc147533520)

[2.7.**(CE-G)** Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones desarrolladas a partir de emuladores. 14](#_Toc147533521)

[2.8 **(CE-H)** Se han empaquetado y desplegado las aplicaciones desarrolladas en dispositivos móviles reales. 14](#_Toc147533522)

[2.9. **(CE-I)** Se han documentado los procesos necesarios para el desarrollo de las aplicaciones. 14](#_Toc147533523)

[Tabla de Ilustraciones 15](#_Toc147533524)

[Bibliografía 15](#_Toc147533525)

# RECURSO DE APRENDIZAJE 1 (RA1)

# (CE-C) Se han identificado las tecnologías de desarrollo de aplicaciones para dispositivos móviles.

Desde la salida del iPhone en junio del 2009, la sociedad ha aumentado el uso de dispositivos móviles, desde casi no tener un móvil por persona en un hogar hasta tener mas de uno por persona, además estos dispositivos han evolucionado y los fabricantes han ido añadiendo mas funcionales que tienen que ser desarrolladas. Para facilitar el desarrollo a los programadores se han creado kit de desarrollos (SDK) según el S.O que usan dichos dispositivos.

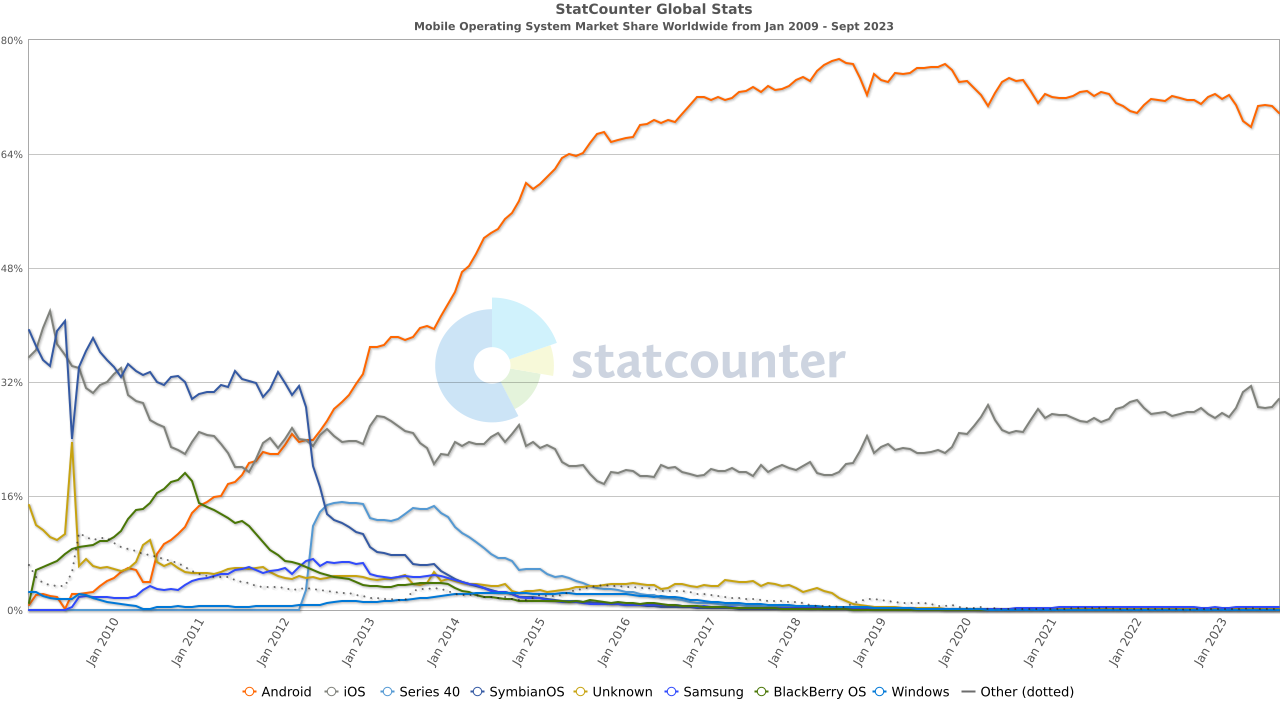
Actualmente el mercado de S.O en dispositivos móviles está dominado en primer lugar por Android (*Propietario Google*) y en segundo lugar IOS (*Propietario Apple*).

Tabla 1 – Estadísticas de uso S.O en dispositivos móviles

El desarrollo para estos dos sistemas está supeditado a los lenguajes de Kotlin (*Oficial de Google para Android desde 2019*), Java (*Lenguaje original para Android y en uso actualmente*) y Swift (*Lenguaje oficial de todos los S.O de Apple*), todos ellos disponen de SDK, el más famoso es el Java Development Kit (*JDK*), por su antigüedad y robustez.

Sobre estos SDK, se han creado frameworks que facilitan aun más el desarrollo de aplicaciones, como por ejemplo React Native o Flutter. Además, si se pretende desarrollar juegos existen los SDK especializados llamados GDK.

# (CE-D) Se han instalado, configurado y utilizado entornos de trabajo para el desarrollo de aplicaciones para dispositivos móviles.

Para manejar los diferentes JDK del apartado 1.1 se han desarrollo entornos de trabajo (*IDEs*) que permiten usar diferentes funcionalidades, por ejemplo, el compilado y ejecución automática del código, autocompletado de código o corrección de errores en tiempo de escritura, entre otras.

Los principales IDEs del mercado son Android Studio para Android y Xcode para IOS.

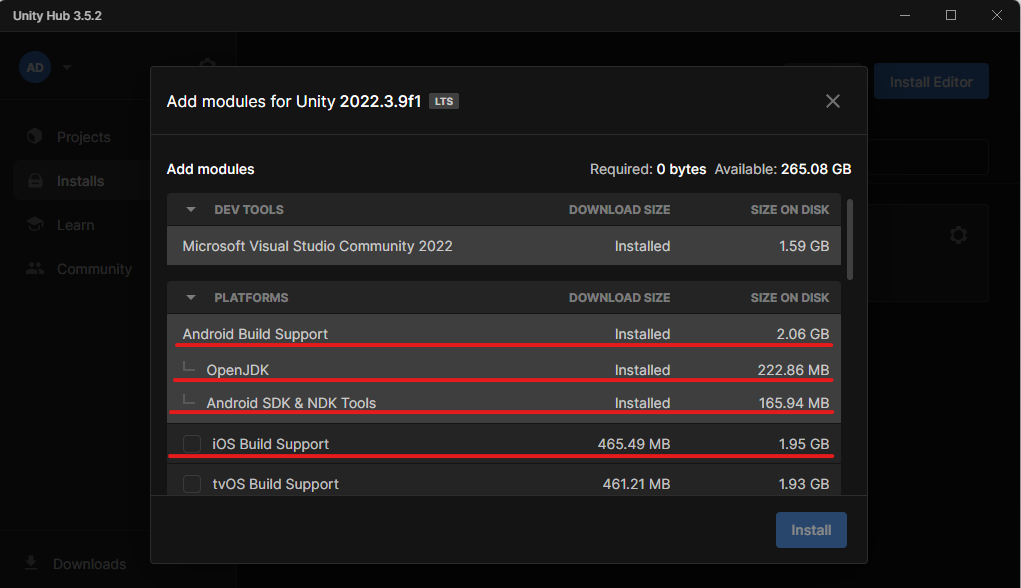
Además, se tiene la posibilidad de usar un motor de videojuegos comercial como Unity, Unreal Engine o Godot para desarrollar en dispositivos móviles añadiendo un módulo extra.

Ilustración 1 – Módulos Unity 2022.3.9f1

# (CE-E) Se han identificado configuraciones que clasifican los dispositivos móviles en base a sus características.

Los dispositivos móviles se clasifican según su propósito y funcionalidad. Aquí te dejo una descripción de cada tipo:

* **Propósito general**: Incluye dispositivos como computadoras, tablets y teléfonos inteligentes avanzados. Se caracterizan por su capacidad para realizar una amplia gama de actividades y sintetizar funciones. Son el medio de trabajo para muchas personas, permitiendo realizar actividades que generan alta productividad.
* **Propósito de entretenimiento**: Este grupo incluye dispositivos diseñados para proporcionar diversión a las personas, por ejemplo, las consolas portátiles (*Nintendo Swtich, Steam Deck o Asus ROG Ally*)

Además, los dispositivos móviles también pueden clasificarse según el usuario objetivo, como móviles para niños y móviles para adultos mayores. También se pueden distinguir por funcionalidades básicas y avanzadas, por ejemplo, los smartphones, tablets o wearable.

# (CE-G) Se ha analizado la estructura de aplicaciones existentes para dispositivos móviles identificando las clases utilizadas.

Al empezar en el desarrollo para dispositivos móviles por el motor grafico Unity, el primer paso es realizar el primer videojuego de la historia (*Pong*), ya que este es el más fácil de programar.

Interfaz de usuario gráfica, Aplicación

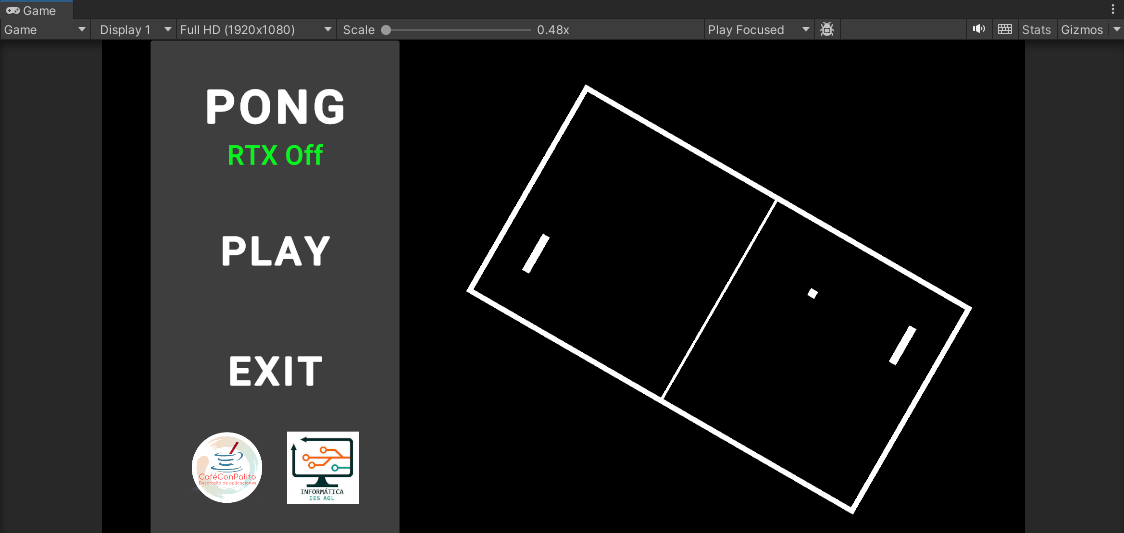
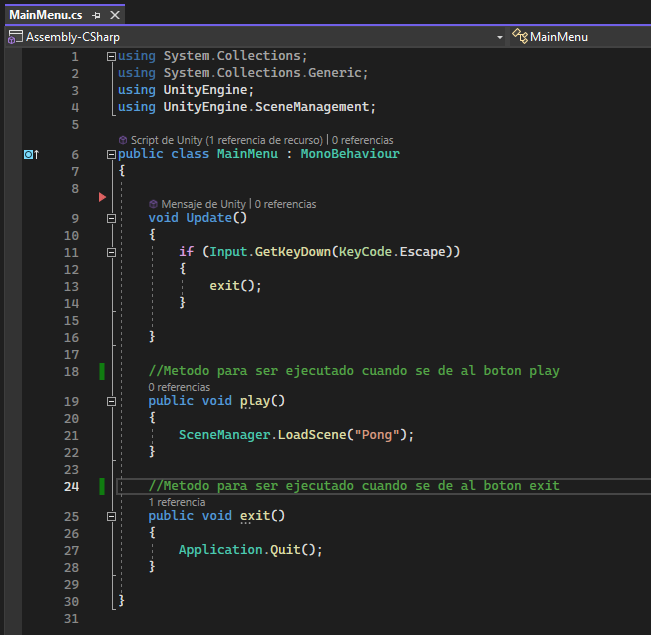
Descripción generada automáticamenteEn Unity se puede realizar con dos clases básicas (*Scripts*) en el lenguaje C#, la primera clase se encarga de calcular el movimiento de la pelota, y la segunda del movimiento de las palas.

Ilustración 2 – Scripts Básicos Pong en Unity

Dichos *Scripts* son agregados en los *GameObjects* correspondientes, ya que si no son agregados nunca serán ejecutados.

# (CE-H) Se han realizado modificaciones sobre aplicaciones existentes.

Al videojuego del Pong clásico se le pueden añadir varias mejoras en la legibilidad del codigo y funciones de videojuegos más nuevos, para ello realice lo siguiente.

Crear un *Main Menu* con las acciones de jugar una partida o salir de la aplicación, para ello se crea un nuevo *script* que controle.

Código 1 – Script Main Menu

Ilustración 3 – Main Menu Pong

Texto

Descripción generada automáticamenteAdemás, el control del juego le centralice en un script llamado GameManager.cs con el patrón de diseño singleton para que solo haya una instancia de esta clase durante el tiempo de ejecución y así eliminar posibles errores de simultaneidad.

Código 2 – Patrón Singleton en Script GameManager.cs

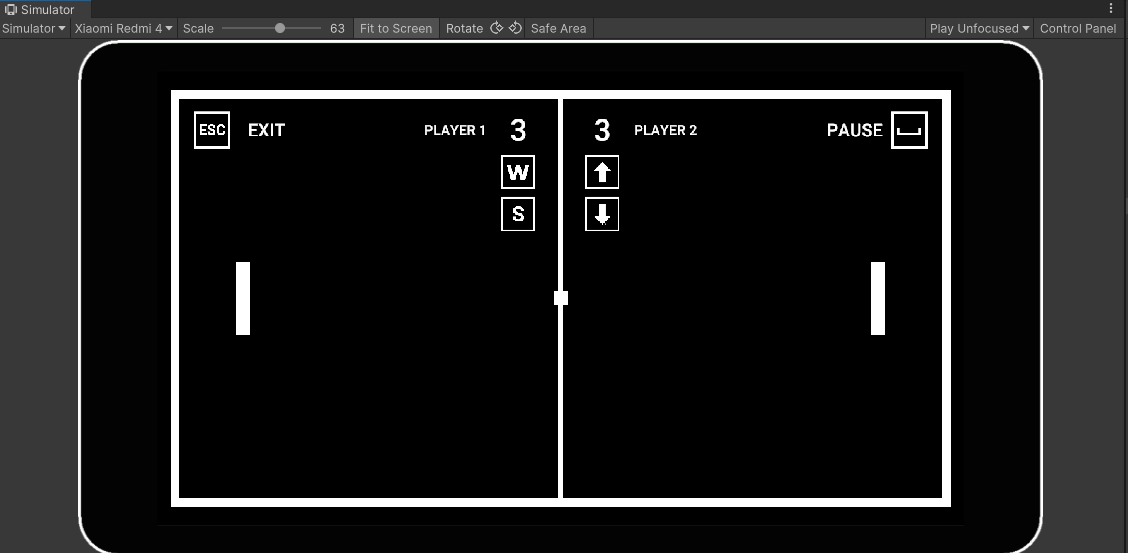
Al centralizar el control del juego en una sola clase y escribir la línea *DontDestroyOnLoad();* nos permite transferir información entre escenas del mismo proyecto sin tener mucha complicación o solo usar un Game Manager para todas las escenas.

# (CE-I) Se han utilizado emuladores para comprobar el funcionamiento de las aplicaciones.

Para comprobar el correcto funcionamiento del juego en diferentes plataformas portátiles se han buscado emuladores para no tener que adquirir el hardware necesario o el kit de desarrollo de los fabricantes.

El más usado para emular el sistema Android es el BlueStacks y para Nintendo Switch es el Yuzu.

Además, hay emuladores para consolas portátiles retro como mGBA para emular la GameBoy Advance en dispositivos con Android o RetroArch que recopila la mayoría de los emuladores de consolas de Nintendo para PC.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamentePero para ser más eficiente me he instalado el paquete *Device Manager* para Unity que me permite probar como seria mi juego en muchos dispositivos móviles sin necesitar hacer un “*Build*”. (*Nota: Desde Unity 2021 ya viene integrado por defecto en las instalaciones y solo amplia el número de dispositivos que se pueden emular*).

Ilustración 4 – Package Manager Unity 2022

Ilustración 5 – Unity Simulando un Móvil

# RECURSO DE APRENDIZAJE 2 (RA2)

# (CE-A) Se ha generado la estructura de clases necesaria para la aplicación.

Texto

Descripción generada automáticamentePara tener un *clean code,* lo mejor es usar el paradigma de programación POO (*Programación Orientada a Objetos*), siguiendo este paradigma Unity crea una estructura de *Game Objects* llamada *Hierarchy*.

Ilustración 6 – Pestaña Hierarchy Unity

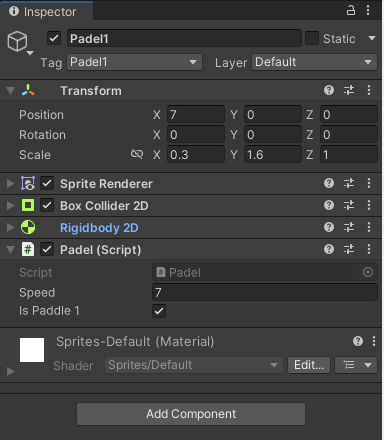
 Además, cada *Game Object* tiene diferentes módulos que definen al objeto o realizan funciones, por ejemplo, el módulo *Transform* define la posición, rotación y escala de Game Object en la escena, o el módulo *Script* determina el codigo que será ejecutado por el Object.

Ilustración 7 – Pestaña Inspector Unity

Diagrama

Descripción generada automáticamentePara el proyecto de Pong se ha generado la siguiente estructura de Scripts y escenas.

Diagrama 1 – Estructura de clases y escenas en el proyecto Pong

# (CE-B) Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamentePara realizar interfaces en Unity, lo primero es crear un *gameObject* de tipo *canvas* que define la dimensión de la interfaz o si es escalable o no, entre otras cosas.

Ilustración 8 – Inspector de un Canvas

El segundo paso es crear un *gameObject* de tipo *EventSystem* para que Unity pueda interactuar con los elementos del canvas, por ejemplo, un botón o un slider.

Si se introduce un elemento UI en la escena con el menú contextual, Unity introduce automáticamente los dos *gameObject* anteriores.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentePara trabajar con las ventanas se pueden usar los elementos *legacy* de Unity, pero lo más común en la actualidad es usar la librería Text Mesh Pro, que brinda más funcionalidades que los elementos antiguos no tienen, como el autosize.

Ilustración 9 – Librería Text Mesh Pro

# (CE-C) Se han utilizado las clases necesarias para la conexión y comunicación con dispositivos inalámbricos.

Unity provee a los desarrolladores una API de alto nivel para la conexión entre dispositivos inalámbricos vía redes de internet, denominada por sus siglas HLAPI. La documentación oficial de Unity la describe de la siguiente forma:

El API de Alto Nivel (HLAPI) es un sistema para construir capacidades multijugador para juegos de Unity. Está construido encima de la capa del nivel menor del transport de la comunicación en tiempo real, y maneja las tareas comunes que son requeridas para juegos multijugador. Mientras que la capa de transporte soporte cualquier tipo de topología de red, el HLAPI es un sistema servidor autoritaria; aunque solo permite que uno de los participantes sean un cliente y el servidor al mismo tiempo, por lo que no hay un proceso dedicado al servidor requerido. Trabajando en conjunto con los servicios de internet, esto permite a que los juegos multijugador sean jugados a través de la internet con el menor trabajo posible de los desarrolladores.

.

# (CE-D) Se han utilizado las clases necesarias para el intercambio de mensajes de texto y multimedia.

# (CE-E) Se han utilizado las clases necesarias para establecer conexiones y comunicaciones HTTP y HTTPS.

En ciertos proyectos es necesario realizar peticiones HTTP a través de los métodos propios (*GET, PUT, POST y DELETE*) para ello Unity creo la clase UnityWebRequest. Para ello te permite crear tres tipos de Objetos.

* Un objeto **UploadHandler** maneja la transmisión de datos al servidor
* Un objeto **DownloadHandler** maneja la recepción, buffering y post-procesamiento de datos recibidos del servidor
* Diagrama

  Descripción generada automáticamenteUn objeto **UnityWebRequest** maneja los otros dos objetos, y también maneja el flujo de control HTTP. Este objeto es dónde los encabezados personalizados y URLs se definen, y dónde la información de error y redireccionamiento se almacena.

Diagrama 2 – Diagrama de Clases UnityWebRequest

Texto

Descripción generada automáticamenteEsta clase nos permite, con pocas líneas de codigo, interactuar con un servidor HTTP, a continuación, muestro un codigo para una petición GET para recuperar un archivo de binarios.

Código 3 – Script para realizar una petición GET

# (CE-F) Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia.

Los proyectos un poco avanzados tienen la necesidad de almacenar datos de forma persistente, para ello en Unity existente tres formas, los *PlayerPrefs*, archivos de binarios y conexión a Bases de Datos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | PlayerPrefs | Archivos Binarios | BBDD |
| Ventajas | Archivos de texto muy simples  Fácil de Usar en Unity | No se pueden manipular de forma simple.  Guardados en local | El SGDB garantiza la integridad y persistencia de los datos |
| Desventajas | Muy vulnerables a cambios, modificaciones o accesos no deseados | Si son muy grandes pueden llegar la ejecución del proyecto al leerlos o escribirlos | Datos almacenados en un servidor, el cual aumenta los costes del proyecto |

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteLos *PlayerPrefs* son fáciles de programar ya que Unity desde la versión 2021 implementa una clase (*PlayerPrefs*) para crear estos tipos de ficheros con unos pocos métodos simples.

Web 1 – Métodos clase PlayerPrefs

Si lo que se desea es crear ficheros de binarios, .net y c# permite la serialización de estos, con las clases del paquete System.Runtime.Serialization, además .net permite la serialización a XML y JSON para transmitir información vía internet.

Si se decide por usar una base de datos relacional, se pueden usar las clases de paquete Microsoft.Data.SqlClient, lo más común es usar el conector del sistema gestor de la propia base de datos usando los tres datos básicos para la conexión (ruta al servidor, base de datos a usar y usuario que realiza la conexión). Además, se puede usar una base de datos NoSQL si las necesidades del proyecto así lo requieren.

# (CE-G) Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones desarrolladas a partir de emuladores.

A lo largo del desarrollo del juego de Pong he realizado diferentes pruebas con mis compañeros para comprobar el correcto funcionamiento en IPhone 13 Pro Max y cuando mis compañeros no podían dedicarme tiempo usaba el emulador de Unity, ya que este me brinda un emulador para el mismo móvil.

Imagen que contiene Diagrama

Descripción generada automáticamenteEl mayor problema que he tenido es que el escalado automático de la interfaz grafica no se ajustaba de forma correcta.

Ilustración 10 – Pestaña Simulator Unity con iPhone 13 Pro Max

# (CE-H) Se han empaquetado y desplegado las aplicaciones desarrolladas en dispositivos móviles reales.

Unity permite la exportación a dispositivos móviles, tanto Android como iPhone, al solo disponer de móviles Android y no tenia la posibilidad de pedir a mis compañeros un iPhone, las pruebas finales las he realizado en un POCO X3 PRO.

Como se dijo en el Apartado 1.2 de este mismo documento, hay que tener los módulos necesarios para poder hacer un “*Build”.*

Interfaz de usuario gráfica

Descripción generada automáticamenteImagen de la pantalla de un celular

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica

Descripción generada automáticamente

Fotografía 1 – APK Instalada

Fotografía 2 – APK Funcionando

Ilustración 11 – Generar APK desde Unity

# (CE-I) Se han documentado los procesos necesarios para el desarrollo de las aplicaciones.

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza mediaAunque para el Pong no se realizó una documentación ya que era un proyecto para aprender a controlar Unity, ya hemos empezado un nuevo proyecto, en el cual se esta siguiendo un proceso de Documentación, el primer paso fue plasmar la idea que teníamos en un folio en sucio.

Fotografía 3 – Folios plasmando idea del Juego

Tras tener la idea plasmada empezamos un GDD (*Game Document Design*) para tener una estructura bien definida para empezar el desarrollo, este documento no es fijo y se ira cambiando y ampliando a lo largo del desarrollo para que todo el desarrollo del proyecto quede plasmado en un documento.

Texto

Descripción generada automáticamente

Ilustración 12 – Instantánea del GDD

# Tabla de Ilustraciones

# Bibliografía

Anon., s.f. *dev.mysql.com/doc/connector-net.* [En línea]   
Available at: https://dev.mysql.com/doc/connector-net/en/connector-net-connections-string.html#connector-net-connections-open

Anon., s.f. *docs.unity3d.com - PlayerPrefs.* [En línea]   
Available at: https://docs.unity3d.com/2021.3/Documentation/ScriptReference/PlayerPrefs.html

Anon., s.f. *docs.unity3d.com - UnityWebRequest.* [En línea]   
Available at: https://docs.unity3d.com/es/2019.4/Manual/UnityWebRequest.html

Anon., s.f. *https://docs.unity3d.com - HLAPI.* [En línea]   
Available at: https://docs.unity3d.com/es/530/Manual/UNetUsingHLAPI.html

Anon., s.f. *learn.microsoft.com - Serialización.* [En línea]   
Available at: https://learn.microsoft.com/es-es/dotnet/standard/serialization/

Anon., s.f. *yuzu-emu.org.* [En línea]   
Available at: https://yuzu-emu.org/

apps.apple.com, 2023. *apps.apple.com.* [En línea]   
Available at: https://apps.apple.com/es/app/xcode/id497799835?mt=12

bluestacks.com, s.f. *bluestacks.com.* [En línea]   
Available at: https://www.bluestacks.com/es/index.html

clasificacionde.org, 2022. *www.clasificacionde.org.* [En línea]   
Available at: https://www.clasificacionde.org/clasificacion-de-dispositivos-moviles/

developer.android.com, 2023. *developer.android.com.* [En línea]   
Available at: https://developer.android.com/studio

godotengine.org, 2023. *godotengine.org.* [En línea]   
Available at: https://godotengine.org/

pccomponentes.com, 2022. *pccomponentes.com.* [En línea]   
Available at: https://www.pccomponentes.com/tipos-de-moviles-y-sus-caracteristicas

spdigitalagency.com, 2023. *spdigitalagency.com.* [En línea]   
Available at: https://spdigitalagency.com/es/blog/popular-frameworks-in-2023-for-mobile-application-development

statcounter.com, 2023. *statcounter.com.* [En línea]   
Available at: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202309

unity.com, s.f. *unity.com.* [En línea]   
Available at: https://unity.com/es

unrealengine.com, s.f. *unrealengine.com.* [En línea]   
Available at: https://www.unrealengine.com/

Wikipedia.org, 2023. *Wikipedia.org - Iphone1.* [En línea]   
Available at: https://es.wikipedia.org/wiki/IPhone\_(1.%C2%AA\_generaci%C3%B3n)

Wikipedia.org, 2023. *Wikipedia.org - Kotlin.* [En línea]   
Available at: https://es.wikipedia.org/wiki/Kotlin\_(lenguaje\_de\_programaci%C3%B3n)#:~:text=El%207%20de%20mayo%20de,alternativa%20al%20compilador%20Java%20est%C3%A1ndar.