



MANUAL DE DESARROLLADOR

Nombre de la Aplicación: AR_APP

Versión del Software: 1.0

Fecha de publicación: 24 de enero de
2025

Desarrollado por: Jinama

Correo de Contacto: jinamadamo@gmail.com

Integrantes: Nawal Bouallala Safyoun,
Jia Le Chen y Marc Martínez Rovira

ÍNDICE

Nº Pág

1. Introducción	2
2. Tecnologías utilizadas	3
3. Configuración Entorno de Desarrollo	4
4. Listado de Scripts	7
5. Listado de Funciones	8
6. Feedback	19
7. Conclusión	20

1. Introducción

1.1. Descripción de la Aplicación

- Esta aplicación es una aplicación diseñada, programada y pensada desde cero a partir de una petición de la empresa Bertrandt.

1.2. Resumen de que hace la aplicación y su propósito

- Nuestra aplicación es una aplicación dedicada a la muestra de modelos en Realidad Aumentada, con los cuales se pueden hacer algunas funcionalidades que más adelante en este mismo manual os explicamos.

1.3. Propósito del Manual

- Este manual ha sido redactado con el propósito de ayudar a todas las personas que quieran ayudar a la modificación y a la mejora de nuestra aplicación.
- Este manual también ha sido redactado con el propósito de hacer que cualquier persona que quiera entender cómo ha sido realizada la programación y la configuración de nuestra aplicación, lo tenga un poco más fácil para entenderlo.

1.4. Público Objetivo

- Esta es una aplicación dedicada a un público que tenga una edad superior a 14 años, que además dispongan de un teléfono móvil con una API que sea compatible con la API de la aplicación.

2. Tecnologías Utilizadas

2.1. Lenguaje de Programación

- El lenguaje de programación utilizado para programar todos y cada uno de los scripts utilizados en la programación de las funcionalidades de nuestra aplicación es C#, llamado C Sharp.

2.2. Unity

- Unity es la plataforma con la que hemos diseñado toda la jerarquía de objetos de nuestra aplicación, es la plataforma con la que hemos diseñado todo el diseño gráfico de la interfaz de nuestra aplicación y es la plataforma con la que hemos hecho todas las pruebas debido a que dispone de un emulador de cualquier tipo de dispositivo con el que hemos podido probar las funcionalidades antes de crear la apk y poder probar la aplicación en un teléfono móvil.

2.3. Repositorio en Github

- Decidimos crear un repositorio en Github para poder tener seguridad en cuanto a que no se perdieran los archivos del proyecto, también para que siempre pudiéramos tener una copia de seguridad del proyecto y para que así pudiéramos realizar todos los cambios que quisieramos de forma local para intentar implementar nuevas funcionalidades y para mejorar las funcionalidades ya implementadas.

2.4. Visual Studio

- Decidimos usar como editor de código el Visual Studio, porque es una aplicación de escritura de código con

muchas funcionalidades que nos permitieron programar de una forma más cómoda y visual.

3. Configuración del Entorno de Desarrollo

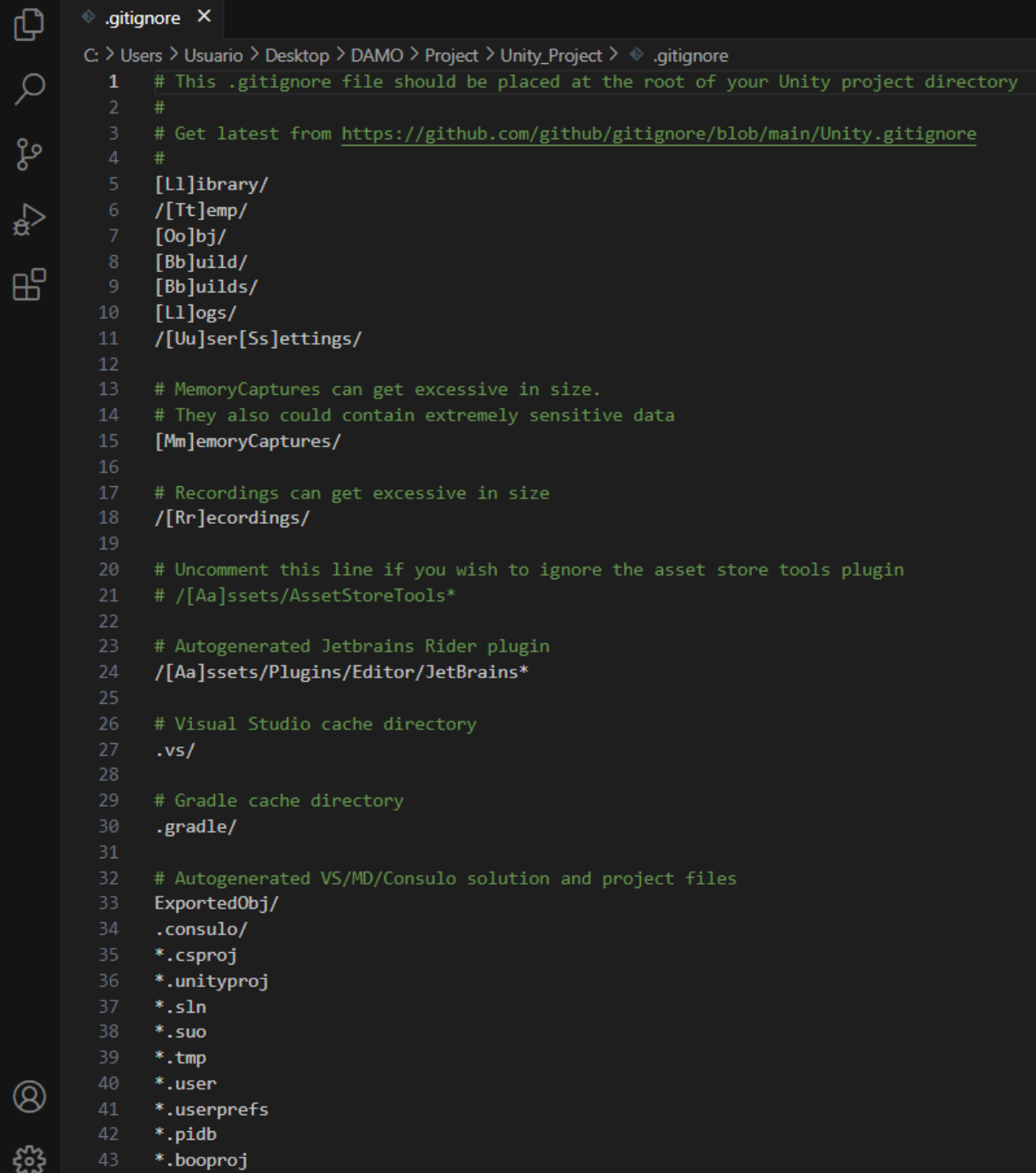
3.1. Configuración del entorno en Unity

- Para poder tener el proyecto bien configurado para utilizarlo con Unity es necesario tener en el apartado File->Build Profiles tienes que tener la plataforma Android descargada y tienes que clickar en esa plataforma y darle al boton Switch Platform para hacer que el proyecto cuando te construya la aplicación para probarla en un dispositivo externo, te construya una apk para probarla en Android.

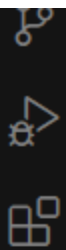
3.2. Como configurar el repositorio de Github para poder hacer push del proyecto de Unity

- Para poder tener un repositorio de Github en el cual poder hacer Push de todos los cambios realizados en nuestro proyecto de Unity, hay que seguir los siguientes pasos para configurarlo:
 - En el momento de crear el repositorio hay que activar la opción de que dentro del repositorio se cree un archivo .gitignore y hay que crearlo usando la plantilla de Unity, porque ese es un archivo que te permite omitir que carpetas quieres hacer que Github no suba en los Push al repositorio para evitar hacer Push con muchos datos innecesarios.

- Esta es la configuración que hay que tener en el .gitignore para poder hacer Push con los datos justos y necesarios de nuestro proyecto:

A screenshot of a code editor with a dark theme. The editor shows the content of a .gitignore file. The file path in the top bar is 'C:\> Users > Usuario > Desktop > DAMO > Project > Unity_Project > .gitignore'. The file content is as follows:

```
1 # This .gitignore file should be placed at the root of your Unity project directory
2 #
3 # Get latest from https://github.com/github/gitignore/blob/main/Unity.gitignore
4 #
5 [Ll]ibrary/
6 /[Tt]emp/
7 [Oo]bj/
8 [Bb]uild/
9 [Bb]uilds/
10 [Ll]ogs/
11 /[Uu]ser[Ss]ettings/
12
13 # MemoryCaptures can get excessive in size.
14 # They also could contain extremely sensitive data
15 [Mm]emoryCaptures/
16
17 # Recordings can get excessive in size
18 /[Rr]ecordings/
19
20 # Uncomment this line if you wish to ignore the asset store tools plugin
21 # /[Aa]ssets/AssetStoreTools*
22
23 # Autogenerated JetBrains Rider plugin
24 /[Aa]ssets/Plugins/Editor/JetBrains*
25
26 # Visual Studio cache directory
27 .vs/
28
29 # Gradle cache directory
30 .gradle/
31
32 # Autogenerated VS/MD/Consulo solution and project files
33 ExportedObj/
34 .consulo/
35 *.csproj
36 *.unityproj
37 *.sln
38 *.suo
39 *.tmp
40 *.user
41 *.userprefs
42 *.pidb
43 *.booproj
```



```
39 *.tmp
40 *.user
41 *.userprefs
42 *.pidb
43 *.booproj
44 *.svd
45 *.pdb
46 *.mdb
47 *.opendb
48 *.VC.db
49
50 # Unity3D generated meta files
51 *.pidb.meta
52 *.pdb.meta
53 *.mdb.meta
54
55 # Unity3D generated file on crash reports
56 sysinfo.txt
57
58 # Builds
59 *.apk
60 *.aab
61 *.unitypackage
62 *.app
63
64 # Crashlytics generated file
65 crashlytics-build.properties
66
67 # Packed Addressables
68 /[Aa]ssets/[Aa]ddressable[Aa]ssets[Dd]ata/*/*.bin*
69
70 # Temporary auto-generated Android Assets
71 /[Aa]ssets/[Ss]treamingAssets/aa.meta
72 /[Aa]ssets/[Ss]treamingAssets/aa/*
73
```

4. Listado de Scripts

- **ARInteractionsManager**
 - Este script tiene como propósito general gestionar la interacción con modelos 3D en un entorno AR.
- **DataManager**
 - Este script gestiona la creación dinámica de botones basados en una lista de elementos (Item). Los botones creados se añaden a un contenedor en la interfaz de usuario (UI) para permitir la interacción del usuario con los elementos
- **DropDownMenuManager**
 - Este script gestiona un menú desplegable en una interfaz de usuario (UI) en Unity. Su objetivo principal es activar y desactivar el menú (mostrarlo u ocultarlo) mediante un sistema de alternancia.
- **GameManager**
 - Este script es el núcleo de la lógica de un sistema en Unity que gestiona el flujo de menús y la interacción con un modelo 3D en una aplicación, posiblemente relacionada con realidad aumentada (AR).
- **Item**
 - Está diseñado para representar datos relacionados con un objeto o ítem dentro de un juego.
- **ItemButtonManager**
 - Este script gestiona la interacción de un botón de ítem en un sistema de interfaz de usuario dentro de Unity. Está diseñado para vincular un botón con un

modelo 3D, configurarlo visualmente en la UI y permitir su interacción en un entorno de realidad aumentada (AR).

- **ModelTouchRotation**
 - Este script está diseñado para controlar la rotación y el movimiento de un modelo 3D en respuesta a las interacciones del usuario a través del mouse o dispositivos táctiles.
- **UIManager**
 -

5. Listado de Funciones

- **ARInteractionsManager**
 - **void Start()**
 - Esta función inicializa el **ARRaycastManager**, que puede ser útil para futuras interacciones con superficies AR.
 - Se registra un evento para limpiar el modelo actual al regresar al menú principal (actualmente comentado).
 - **public void ResetCameraPosition()**
 - Reinicia la posición de la cámara a (0, 0, 0) y opcionalmente su rotación. Esto podría usarse para centrar la cámara en la escena.
 - **private void ResetItemPosition()**

- Elimina el modelo actual si no ha sido aprobado.
- public void ApproveModel()
 - Marca el modelo actual como aprobado.
 - Redirige al menú principal de la aplicación.
- public void DeleteModel()
 - Elimina el modelo activo de la escena y resetea la bandera isModelApproved.
- public void DeleteAllModels()
 - Encuentra y elimina todos los modelos con la etiqueta "InstantiatedModel".
 - Resetea el estado de aprobación.
- DataManager
 - void Start()
 - Se suscribe al evento OnItemsMenu del GameManager.
 - Este evento, cuando se dispara, ejecuta el método CreateButtons() para generar los botones correspondientes.
 - private void CreateButtons()
 - Iterar sobre la lista de elementos (Item):
 - Por cada Item en la lista, crea dinámicamente un botón basado en el prefab itemButtonManager.
 - Asigna el botón al buttonContainer con su padre.
 - Asignar propiedades al botón:

- Para cada botón creado, asigna las propiedades del Item correspondiente:
 - ItemName: El nombre del elemento.
 - ItemDescription: La descripción del elemento.
 - ItemImage: La imagen asociada al elemento.
 - Item3DModel: El modelo 3D del elemento.
- Nombrar el botón:
 - El nombre del botón se establece como el nombre del elemento (item.ItemName) para facilitar su identificación.
- Depuración:
 - Utiliza Debug.Log para imprimir las propiedades de cada elemento en la consola. Esto es útil para verificar que los datos se asignaron correctamente.
- Desuscribirse del evento:
 - Una vez que se crean los botones, el método se desuscribe del evento OnItemsMenu para evitar duplicaciones si se vuelve a abrir el menú.
- DropdownMenuManager
 - void Start()
 - Este método se ejecuta al inicio de la escena.
 - Objetivo: Asegurarse de que el menú esté oculto al comenzar:
 - Si el dropdownMenu está asignado, lo desactiva para que no sea visible.

- `public void ToggleMenu()`
 - Verifica si el `dropdownMenu` no es `null`. Si no está asignado, muestra un error en la consola usando `Debug.LogError`.
 - Invierte el estado de `isMenuActive`:
 - Si el menú está oculto, lo activa (`true`).
 - Si el menú está visible, lo oculta (`false`).
 - Activa o desactiva el `dropdownMenu` basado en el valor de `isMenuActive`:
 - Imprime un mensaje en la consola indicando si el menú está abierto o cerrado:
- **GameManager**
 - `private void Awake()`
 - Configura el `GameManager` como un Singleton.
 - Si ya existe una instancia, destruye la nueva para evitar duplicados.
 - `void Start()`
 - Verifica si existe la etiqueta `InstantiatedModel` en el proyecto.
 - Activa el menú principal mediante `MainMenu()`.
 - `public void MainMenu()`
 - Activa eventos asociados al Main Menu para notificar a otros scripts.
 - Imprime un mensaje en la consola indicando el estado actual.
 - `public void ItemsMenu()`
 - Activa eventos asociados al Items Menu para notificar a otros scripts.

- Imprime un mensaje en la consola indicando el estado actual.
- `public void ARPosition()`
 - Activa eventos asociados a la posición en Realidad Aumentada para notificar a otros scripts.
 - Imprime un mensaje en la consola indicando el estado actual.
- `public void CloseAPP()`
 - Esta función cierra la aplicación cuando se llama.
 - Es útil en aplicaciones que necesitan un botón para salir del programa, como un botón "Salir" en el menú principal.
- `public void ApproveModel()`
 - Marca un modelo 3D como "aprobado" en el flujo lógico del juego.
 - Luego de aprobar el modelo, redirige al usuario al menú principal llamando al método `MainMenu()`.
- `public void DeleteCurrentModel()`
 - Elimina el modelo actual notificando a un script externo (`ARInteractionsManager`).
 - Retorna al menú principal.
- `public void Closemodels()`
 - Elimina todos los modelos visibles usando el `ARInteractionsManager`.
 - Retorna al menú principal.
- `private bool TagExists(string tag)`

- Tiene como propósito verificar si una etiqueta (tag) específica existe en el proyecto de Unity.
- public void SetCurrentModel(GameObject model)
 - Define el modelo actualmente seleccionado.
 - Deselecciona el modelo anterior (si existe).
 - Almacena los colores originales de los renderizadores del nuevo modelo.
- private void DeselectCurrentModel()
 - Deselecciona el modelo actual restaurando sus colores originales.
 - Opcionalmente, puede desactivar, destruir o mover fuera de la pantalla el modelo.
- public void ChangeModelColorToRed()
 - Cambian el color del modelo actual a rojo.
 - Verifican si el modelo tiene componentes `Renderer` con la propiedad `_Color`.
- public void ChangeModelColorToBlue()
 - Cambian el color del modelo actual a azul.
 - Verifican si el modelo tiene componentes `Renderer` con la propiedad `_Color`.
- public void ChangeModelColorToLila()
 - Cambian el color del modelo actual a lila.
 - Verifican si el modelo tiene componentes `Renderer` con la propiedad `_Color`.
- public void ChangeModelColorToPurple()
 - Llama a la función `ChangeModelColorToLila()`
- public void ChangeModelColorToDefault()

- Restaura los colores originales del modelo usando el diccionario originalColors.
- public void EnableMoveMode()
 - Habilitan los modos de mover en el modelo usando el componente ModelTouchRotation.
- public void EnableRotateMode()
 - Habilitan los modos de rotar en el modelo usando el componente ModelTouchRotation.
- public void DisableAllModes()
 - Desactiva todos los modos de interacción del modelo.
- public void EnlargeModel()
 - Incrementa el tamaño del modelo en un 20%.
 - Asegura que el tamaño no exceda un límite definido (15 unidades).
- public void ShrinkModel()
 - Reduce el tamaño del modelo en un 20%.
 - Asegura que el tamaño no sea menor que un límite mínimo (0.00001 unidades).
- public void RotateModelClockwise()
 - Rota el modelo 15 grados en sentido horario (eje Y).
- public void RotateModelCounterClockwise()
 - Rota el modelo 15 grados en sentido antihorario (eje Y).

- Item

- ItemButtonManager
 - void Start()
 - Configuración del texto del nombre del ítem:
 - Busca un componente TextMeshProUGUI en el segundo hijo del objeto actual y asigna el itemName como texto. Si no encuentra el componente, lanza un error.
 - Configuración de la imagen del ítem:
 - Busca un componente RawImage en el primer hijo del objeto actual y asigna la textura de itemImage. Si la imagen es nula o el componente no existe, muestra advertencias o errores.
 - Configuración del botón:
 - Asocia el método GameManager.instance.ARPosition y el método Create3DModel como acciones al hacer clic en el botón.
 - Busca en la escena el componente ARInteractionsManager usando FindAnyObjectByType. Si no lo encuentra, lanza un error.
 - private void Create3DModel()
 - Validación del modelo y del gestor de interacciones:
 - Comprueba si item3DModel y interactionsManager no son nulos.
 - Instanciación del modelo 3D:
 - Crea una copia (Instantiate) de item3DModel.

- Le asigna la etiqueta "InstantiatedModel" para facilitar su identificación.
 - Añade el componente ModelTouchRotation al modelo para habilitar interacciones (como mover o rotar el modelo).
 - Configuración del modelo en el gestor de interacciones:
 - Asigna la instancia del modelo al gestor ARInteractionsManager para que maneje su posicionamiento y lógica en el entorno AR.
- ModelTouchRotation
 - private void Update()
 - Escucha continuamente las entradas del usuario:
 - Si el botón izquierdo del mouse está presionado:
 - Registra la posición actual del mouse y calcula el desplazamiento (delta).
 - Aplica el desplazamiento para mover o rotar el modelo según el modo habilitado.
 - Si una pantalla táctil está presente:
 - Registra y procesa las posiciones táctiles de forma similar al mouse.
 - private void RotateModelYAxis()
 - Rota el modelo alrededor del eje Y a una velocidad constante definida por autoRotationSpeed.

- `public void MoveModel(Vector3 deltaPosition)`
 - Desplaza el modelo en el espacio mundial según el vector `deltaPosition`.
- `public void RotateModel(float deltaX)`
 - Convierte las coordenadas del mouse de la pantalla al espacio mundial y mueve el modelo a esa posición.
- `public void MoveModelToMouse(Vector2 mousePosition)`
 - Rota el modelo en el eje Y según el desplazamiento horizontal del mouse (`deltaX`).
- `public void EnableMoveMode()`
 - Controla si el modelo puede moverse.
- `public void EnableRotateMode()`
 - Controla si el modelo puede rotar.
- `public void DisableAllModes()`
 - Controla si el modelo tiene las funcionalidades de moverse y de rotar deshabilitadas.
- **UIManager**
 - `void Start()`
 - El UIManager se suscribe a eventos que el GameManager emite:
 - OnMainMenu: Evento para activar el menú principal.
 - OnItemsMenu: Evento para activar el menú de ítems.

- OnARPosition: Evento para activar el menú de posicionamiento en realidad aumentada (AR).
- Cuando uno de estos eventos se emite, se ejecuta la función asociada.
- private void ActivateMainMenu()
 - Menú principal:
 - Los hijos del mainMenuCanvas se escalan a su tamaño completo (1, 1, 1) con animaciones de 0.3 segundos.
 - Menú de ítems:
 - Los hijos del itemsMenuCanvas se escalan a (0, 0, 0) (desaparecen).
 - Uno de los hijos también cambia su posición en el eje Y a 180 en 0.3 segundos.
 - Menú AR:
 - Los hijos del ARPositionCanvas se escalan a (0, 0, 0) (desaparecen).
- private void ActivateItemsMenu()
 - Menú principal:
 - Los hijos del mainMenuCanvas desaparecen escalándose a (0, 0, 0).
 - Menú de ítems:
 - Los hijos del itemsMenuCanvas se escalan a (1, 1, 1) (aparecen).
 - Uno de los hijos también cambia su posición en el eje Y a 300.
- private void ActivateARPosition()
 - Similar a las funciones anteriores, pero esta vez:

- Los hijos del ARPositionCanvas aparecen escalándose a (1, 1, 1).
- Los menús principal e ítems desaparecen.

6. Feedback

6.1. Para cualquier problema o error que encuentren en la aplicación póngase en contacto con los desarrolladores a través de los siguientes correos electrónicos:

- jinamadamo@gmail.com
- nawal.bouallala@estudiantat.upc.edu
- jiale.chen@estudiantat.upc.edu
- marc.martinez.rovira@estudiantat.upc.edu

6.2. Formato para informar de un problema o error

- Redactar un correo electrónico a alguno de los correos electrónicos mencionados anteriormente con el siguiente formato:
 - Asunto: Explicar el error o problema te ha ocurrido de forma muy resumida.
 - Cuerpo: Explicar detalladamente qué error o problema te ha ocurrido, explicitando que mensaje de error te ha salido, explicitando utilizando qué funcionalidad te ha ocurrido y explicando todos los detalles posibles que hagan referencia al error o problema ocurrido.

6.3. Para cualquier sugerencia o mejora que quieran aportar a la aplicación póngase en contacto con los desarrolladores a través de los siguientes correos electrónicos:

- jinamadamo@gmail.com
- nawal.bouallala@estudiantat.upc.edu
- jjale.chen@estudiantat.upc.edu
- marc.martinez.rovira@estudiantat.upc.edu

6.4. Formato para informar de una sugerencia o mejora

- Redactar un correo electrónico a alguno de los correos electrónicos mencionados anteriormente con el siguiente formato:
 - Asunto: Explicar la sugerencia o mejora que se te ha ocurrido de forma muy resumida.
 - Cuerpo: Explicar detalladamente qué sugerencia o mejora se te ha ocurrido, explicando todos los detalles posibles que hagan referencia a la mejora o sugerencia que se te ha ocurrido para nuestra aplicación.

7. Conclusión

7.1. Agradecimientos

- Te agradecemos que hayas decidido tomarte el tiempo para descargarte e instalarte nuestra aplicación.
- Sabemos que nuestra aplicación está en su primera versión y seguramente tengas muchas

ideas para mejorarla, por lo que te animamos a realizar correos electrónicos siguiendo el formato mencionado anteriormente para poder aportar esas grandes ideas a versiones futuras de nuestra aplicación.