Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

PROTOTIPO de APP de REALIDAD AUMENTADA para ANDROID

Autor: Christian Paniagua Paniagua
Tutor(a): Ángel Lucas González Martínez

Madrid, junio 2021

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: PROTOTIPO de APP de REALIDAD AUMENTADA para ANDROID
Junio 2021

Autor: Christian Paniagua Paniagua

Tutor:
Ángel Lucas González Martínez
DLSIIS
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este proyecto consiste en realizar un prototipo basado en RA/AR (Realidad Aumentada/ *Augmented Reality*) de uno de los edificios de la ETSIINF, UPM, utilizando la plataforma Unity, para dispositivos *Android*.

El objetivo es recrear dicho edifico en 3D, poder verlo con la cámara del dispositivo, al enfocar a un marcador específico, impreso en papel, con un diseño adecuado y tener la posibilidad de interactuar con el modelo, para poder ver cada una de sus plantas, despachos, aulas… y que se ofrezca cierta información de cada lugar. Además, el edificio se podrá ver desde cualquier perspectiva, así como acercar y alejar la cámara, del dispositivo, para obtener más o menos detalle gráfico, respectivamente.

Los objetivos de este proyecto son:

- Estudio de las capacidades para trabajar con RA, que proporciona Unity, para dispositivos *Android*.
- Determinar los requisitos mínimos de *HW* y de *SW*, de los dispositivos, en los que se usará la aplicación.
- Búsqueda y selección de las herramientas, librerías y *SDKs* adecuadas, que permitan el desarrollo del proyecto, en Unity.
- Definir la información que necesita un modelo 3D para poder implementarse en la herramienta.
- Definir las funcionalidades mínimas de la aplicación.
- Seleccionar un edificio de la facultad (en caso de que no diese tiempo, elegir al menos una planta de ese edificio), para recrearlo en 3D, y poder visualizarlo en RA, así como crear su respectivo marcador, impresa en papel, para que la cámara del dispositivo pueda reconocerlo.
- Si hay tiempo, se creará un modelo en 3D de otro edificio de la facultad y se proporcionará la información para implementarlo.

Abstract

This project is about making a prototype based on AR (Augmented Reality) of one of the buildings of the ETSIINF, UPM, using the Unity platform, for Android devices.

The goal is to model this building in 3D and be able to observe it with the device's camera, when it focuses on a specific marker. The marker will be printed on paper, with a suitable design. In addition, you will be able to "touch" the 3D model and observe each of its floors with its respective offices, classrooms, among others (these places will have panels with relevant information). The 3D model can be seen from any perspective, and you will be able to zoom in and zoom out of the device, to obtain more or less graphic details, respectively.

The goals of this project are:

- Study the capabilities to work with AR, provided by Unity, for Android devices.
- Determine the minimum requirements of HW and SW, of the devices, in which the application will be used.
- Search and select the appropriate tools, libraries and SDKs that allow the development of the project, in Unity.
- Define the information that a 3D model needs to be able to be implemented in the tool.
- Define the minimum functionalities of the application.
- Select a building of the faculty, model it in 3D and observe it in AR. In the event of not having enough time, choose at least one floor of the building. In addition, create its respective marker, printed on paper, and that the device's camera recognize it.
- If there is time, another building of the faculty will be modelled in 3D and the necessary information will be provided to implement it in the application.

Tabla de contenidos

Tabla de figuras

1 Introducción

1.1 Introducción a la realidad aumentada

La realidad aumentada o RA consiste en un conjunto de tecnologías que permiten visualizar, a través de un dispositivo, el mundo real con información gráfica añadida sobre los datos físicos. [1]

Figura 1 Ejemplo de realidad aumentada. DMEXCO (2018). [2]

Este proyecto está dirigido hacia dispositivos móviles con un sistema operativo basado en *Android*, por lo que la información que se muestra acerca del desarrollo se limita a este sector compatible con la RA.

En relación con la tecnología usada, los elementos o componentes, del dispositivo, que intervienen en hacer posible la realidad aumentada son:

- **Cámara:** se encarga de obtener la información del mundo real, la cual será utilizada para captar ciertos elementos físicos, que activen información virtual, denominados activadores. [1] Algunos de estos activadores son:
    - Código QR. [1] Véase un ejemplo de este tipo de activador en la figura mostrada a continuación:

Figura 2 Ejemplo de código QR que dice: "Ojalá vivas tiempos interesantes". *Rotondas, T. (2008).* [3]

- Marcadores. [1] En la siguiente figura se puede observar un ejemplo de un marcador muy simple:

Figura 3 Ejemplo de marcador. Elaboración propia (2020).

- Imagen, objeto o señal GPS. [1] A continuación, se puede observar un ejemplo de un objeto, fácil de reconocer por su color y forma:

Figura 4 Ejemplo de objeto fácil de reconocer mediante escaneo y aplicar características de realidad aumentada. PNG Play (2019) [4]

- **Procesador:** su función es la de superponer la información virtual con la que se obtiene mediante la cámara. [1]
- **Pantalla:** en este componente, del dispositivo, es posible la visualización de la superposición de la información virtual y la real. [1]
- **Software:** es necesario una aplicación que gestione este proceso. [1]

La realidad aumentada posee diferentes niveles, en concreto cuatro, y cada uno de ellos posee unas características, que son:

- **Nivel 0:** consiste en analizar un código, el cual es simplemente un diseño 2D, y enlazar a otro contenido. [1], [5]

  En la figura, que se muestra a continuación, se puede ver un ejemplo de escaneo de un código QR:

Figura 5 Ejemplo de escaneo de código QR. Lang A. (2019). [6]

- **Nivel 1:** este nivel se basa en reconocer un marcador, es decir, una imagen o diseño, en 2D, blanco y negro (debido al alto contraste, ya que el dispositivo lo captará con mayor facilidad), con cualquier tipo de patrón formado por formas geométricas, las cuales, en la mayoría de los casos son rectangulares o cuadradas. Una vez reconocido, se busca la representación de modelos 3D. [1], [5]

  Véase en la siguiente figura como a partir de un marcador, con un diseño fácil de reconocer, se genera un modelo 3D encima de este de forma virtual:

Figura 6 Ejemplo de interacción con marcador. El Filali, Y. & Salah-ddine, K. (2019). [7]

- **Nivel 2:** no utiliza marcadores [1], [5], pero puede hacer uso de:
    - Brújula, también llamada magnetómetro: para conocer la orientación de donde apunta la cámara. [1], [5]
    - GPS**:** para establecer la localización. [1], [5]

  Gracias a estos componentes se puedan usar puntos de interés que muestren información. [1], [5]

  Además, también se pueden reconocer superficies, objetos e imágenes, de tal manera que se puede superponer información gráfica virtual, en tiempo real, sobre estos y que permanezca ahí. [1], [5]

  A continuación, se muestra, en la figura, un ejemplo del nivel 2 de la realidad aumentada, en el que se puede observar cómo escaneando un objeto activador, se genera cierta información virtual:

Figura 7 Ejemplo escaneo de objeto y generación de contenido virtual. PIX4D (2019). [8]

- **Nivel 3:** este nivel es el menos explotado y se basa en el uso de dispositivos como las gafas *Google Glass*, que ofrecerán una experiencia más inmersiva. [1], [5]

  Véase, en la figura mostrada, un ejemplo de un desarrollo que permitiría la realidad aumentada de nivel 3:

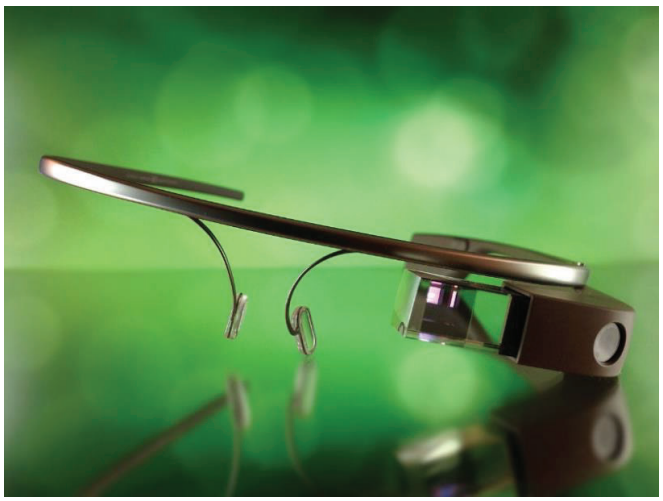

Figura 8 Google Glass. Dan Leveille (2014). [9]

Se pueden encontrar softwares que permiten la realidad aumentada, algunos de ellos son:

- **Unity:** es un motor de desarrollo de videojuegos. En el siguiente enlace se puede encontrar la página oficial. [1], [10], [11]

- **Blender:** es un software que tiene como funcionalidades el modelado, renderizado, animación, etc. En el siguiente enlace se puede encontrar la página oficial. [1], [12]

Por otro lado, algunas de las plataformas que permiten que la realidad aumentada pueda ser experimentada son:

- ***ARCore* (Google)**: es una herramienta de software o tecnología, desarrollada por Google, que permite la creación de aplicaciones basadas en realidad aumentada. Se puede encontrar la página oficial en el siguiente enlace. [1], [13], [14], [19]

- ***ARKit* (Apple):** es una herramienta que permite crear aplicaciones de realidad aumentada, compatibles con productos de Apple: por ejemplo: *iPhone*. Se puede encontrar la página oficial en el siguiente enlace. [1], [15]

- ***Viur***: es una plataforma para realidad aumentada. Se puede encontrar la página oficial en el siguiente enlace. [1], [16]

Por último, como información introductoria, la realidad aumentada puede utilizarse como método educativo, enfocado al turismo, para proporcionar información, medicina, entretenimiento, etc. [1]

1.2 Objetivos del proyecto

Los objetivos que se alcanzarán en este proyecto son:

- Estudio de las capacidades para trabajar con RA, que proporciona Unity, para dispositivos *Android*.

- Determinar los requisitos mínimos de *HW* y de *SW*, de los dispositivos, en los que se usará la aplicación.

- Búsqueda y selección de las herramientas, librerías y *SDKs* adecuadas, que permitan el desarrollo del proyecto, en Unity.

- Definir la información que necesita un modelo 3D para poder implementarse en la herramienta.

- Definir las funcionalidades mínimas de la aplicación.

- Seleccionar un edificio de la facultad (en caso de que no diese tiempo, elegir al menos una planta de ese edificio), para recrearlo en 3D, y poder visualizarlo en RA, así como crear su respectivo marcador, impresa en papel, para que la cámara del dispositivo pueda reconocerlo.

- Si hay tiempo, se creará un modelo en 3D de otro edificio de la facultad y se proporcionará la información para implementarlo.

2 Desarrollo

2.1 Capacidades RA de Unity para *Android*

Unity es una herramienta o *framework* que permite crear un entorno interactivo para diferentes plataformas, como, por ejemplo: dispositivos móviles, PC, etc. [10], [11] Por lo que, en relación con dispositivos móviles, Unity tiene la capacidad de poder crear desarrollos compatibles con los que tengan un sistema operativo *Android*, entre otros. Por lo tanto, es posible generar aplicaciones o juegos, basados en realidad aumentada, para estos, siempre y cuando sean compatibles.

En relación con la realidad aumentada, en *Android*, Unity permite desarrollar aplicaciones que tengan está característica gracias a la tecnología *ARCore*, que proporciona Google, la cual ofrece dos *SDKs*: *AR Foundation* y *ARCore SDK for Unity*. [17]

2.2 Requisitos mínimos hardware y software

Los requisitos mínimos de hardware de los dispositivos móviles *Android* para que puedan hacer uso de aplicaciones con realidad aumentada, pueden variar mucho, debido a que los fabricantes, de los dispositivos, son diferentes y usan distintos componentes con mayor o menor potencia. [19], [20] Por tanto, los requisitos mínimos hardware estarán determinados por los *SDKs*, librerías, paquetes, herramientas, etc. que se usen, es decir, variará en base a qué se utilice.

Teniendo en cuenta lo anteriormente mencionado, Google facilita el trabajo y proporciona una plataforma, denominada *ARCore* (más adelante se explicará el motivo por el cual se eligió el uso de esta plataforma) y la lista oficial de dispositivos compatibles con esta, en el siguiente enlace. [1], [13], [14], [18], [19]

Por otro lado, en relación con los requisitos mínimos de software, aparte de que el dispositivo se encuentre en la lista, debe tener una versión igual o superior a *Android 7.0* (*Nougat*). [19]

2.3 Información asociada a modelo y fichero

Los modelos de cada edificio serán creados mediante un software determinado, cada edificio deberá tener unos requisitos asignados, es decir, unas medidas determinadas, colores, diseño, etc. para cumplir el estándar de la APP.

Cada modelo tiene un marcador, o plantilla, asignado, con un diseño especifico, en 2D, el cual ya estará incorporado en la aplicación para que simplemente sea crear el modelo e incorporarlo en el editor.

Para la muestra de información, se fijará, en el modelo 3D, texto informativo en cada zona de interés, de tal manera que sea visible en la pantalla del dispositivo. Se ha optado por esta opción y no por la dinámica (una versión basada en puntos de interacción, es decir, que la cámara sea capaz de detectarlos y generen información), debido a la dificultad y falta de tiempo para desarrollar esa funcionalidad de la aplicación.

Teniendo en cuenta lo mencionado anteriormente, se deberá escanear el marcador y este deberá ser reconocido por la aplicación. Seguidamente, la aplicación deberá acceder a un registro de datos que contendrá las relaciones entre los marcadores y modelos 3D, para así poder extraer la información de este.

Haciendo un inciso en relación con esto último, y entrando en más detalle acerca del registro de datos, se usará un objeto, sin forma 3D, para cada bloque (cada uno con un nombre específico, ya que deberá ser igual que el del marcador al que irá asociado. Además, agrupará, en forma de hijos, el modelo 3D de las diferentes plantas de este), e irán almacenados en una lista para obtenerlos mediante *scripts*. Una vez definidos los objetos que contendrán los modelos 3D, los marcadores, como ya se ha especificado con anterioridad, deberán tener el mismo nombre que el bloque que tiene asociado e irán almacenados en una lista. Para un correcto funcionamiento, se usarán diccionarios, que permitirán crear una relación entre el modelo 3D y el marcador. Todo lo mencionado con anterioridad se explicará, más en detalle, en la sección que trata sobre el diseño y la implementación de la solución.

Dejando a un lado el inciso, y retomando el punto en el que se extrae la información del modelo 3D, se da paso a la generación de este, permitiendo un uso interactivo para poder ver cada capa del edificio en cuestión.

A continuación, se muestra un diagrama que, de manera simplificada, muestra el orden de procesos:

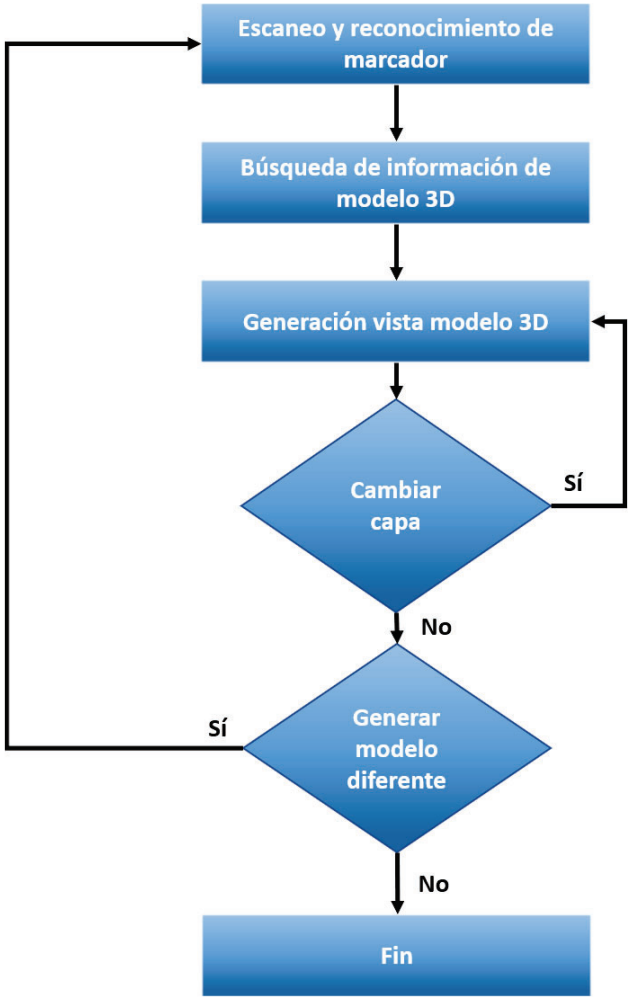

Figura 9 Diagrama de procesos de la aplicación. Elaboración propia (2020).

2.4 Funcionalidades de la *APP*

Las funcionalidades que se desarrollarán para la aplicación serán, ordenadas en orden de prioridad, las siguientes:

1. **Escaneo y reconocimiento correcto del marcador, con diseño específico para cada modelo:** El marcador tendrá un diseño 2D específico, en blanco y negro, de elaboración propia.
2. **Generación del modelo 3D tras el escaneo:** una vez que el paso anterior funcione correctamente, se deberá activar la generación del objeto 3D, para ello, se deberá obtener la información de dicho modelo, de un registro, y representarla.
3. **Visualización correcta del modelo en 3D:** las dimensiones deberán ser correctas a la hora de generarlo, y se realizarán pruebas para comprobarlo, ya que Unity no trabaja con una escala fija, es decir, varios objetos diferentes pueden tener medidas diferentes y tener tamaños similares.
4. **Selección de capas:** la aplicación debe permitir interactuar con un elemento visible en la interfaz, mostrado en pantalla, y ser capaz de ir mostrando las diferentes plantas del edificio, cada vez que recibe un evento.
5. **Mostrar identificador (nombre o número) del aula, despacho, etc.:** se asignará la información de manera fija al modelo y será visible sin ningún tipo de interacción.
6. **Pequeña guía de uso o tutorial:** se creará una interfaz simple, con una pequeña guía de instrucciones y se podrá visualizar al interactuar con un elemento visible en la pantalla del dispositivo.

2.5 *SDK* RA Unity, paquetes, versiones y herramientas necesarias

Como se mencionó con anterioridad, en este proyecto se hará uso de la plataforma *ARCore* que Google implementa en sus dispositivos *Android*. Esto se debe a que esta plataforma garantiza o certifica un correcto funcionamiento, debido a la comprobación de la calidad de cámara, sensores de movimiento, arquitectura de diseño y, además, una potencia de CPU suficiente, para así poder conseguir un rendimiento óptimo y unos cálculos efectivos en tiempo real. [18], [19]

Además, otro de los motivos por los que se hará uso de esta plataforma es debido a que se puede usar en *IOS* y *Android*, a diferencia de *ARKit,* que solo es compatible con *IOS*.

Por otro lado, *ARCore* no precisa de licencia, como ocurre con *Vuforia Engine*.

Como último motivo, se usará el *SDK AR Foundation* y *ARCore XR Plugin* y deben funcionar sobre la plataforma *ARCore,* debido a que el proyecto va enfocado a *Android.*

La elección de *SDKs, softwares,* paquetes…, debe enfocarse en que sean de uso gratuito, al menos si el fin no es comercial, como es el caso de este proyecto. Para la posible elaboración del proyecto en Unity, será necesario lo siguiente:

- ***SDK AR Foundation 4.1.1***: es un *framework,* de alto nivel, diseñado para el desarrollo de la realidad aumentada, en Unity. Además, es multiplataforma, ya que soporta otras tecnologías, las cuales son: *ARCore XR Plugin* para *Android*, *ARKit XR Plugin* para *IOS*, *Magic Leap XR* para *Magic Leap* y *Windows XR Plugin* para *HoloLens*. Este *framework,* en este proyecto, se apoya en el paquete *ARCore XR Plugin* y de forma opcional en *ARCore Extensions.* [17], [22], [23]

  Se ha optado por este *SDK* y no otro como *ARCore SDK,* que es el otro *SDK* que proporciona la plataforma *ARCore*, debido a que es multiplataforma y en caso de querer migrarlo, a otro sistema operativo, como *IOS,* no habría ningún tipo de problema. Además, posee funcionalidades adicionales de Unity, posee un amplio soporte, muchos ejemplos de proyectos y gratuito. En relación con esta última característica mencionada, ya no se podría optar por *Vuforia*, debido a que este *SDK* precisa de licencia, por lo que su uso queda excluido de este proyecto.

- ***ARCore XR Plugin 4.1.1:*** es un paquete que permite activar o habilitar la compatibilidad con *ARCore* mediante la *API XR* multiplataforma de Unity. En este caso, no hay opciones para comparar y valorar que paquete usar y cual no, debido a que este es del que depende el *SDK AR Foundation* para *Android.*

  Por otro lado, implementa los siguientes subsistemas *XR* [17], [24-35] (cada uno maneja una funcionalidad diferente):

  - *Session*: se trata de una instancia de RA, permite que los demás subsistemas funcionen correctamente. Si por algún casual, este se detiene o hay algún tipo de fallo en la creación, no funcionarán.

  - *Camera*: permite manejar la cámara del dispositivo.

  - *Depth:* puede detectar la profundidad de la escena.

  - *Input*: puede habilitar y deshabilitar los *inputs* procedentes de un *plugin* específico.

  - *Planes*: permite detectar superficies planas en el entorno.

  - *Raycast*: añade la posibilidad de enviar “rayos” contra elementos en RA.

  - *Anchors*: permite fijar o anclar la posición de elementos, en RA, en el entorno.

  - *Face Tracking:* reconocimiento y seguimiento de rostros.

  - *Image Tracking:* reconocimiento y seguimiento de imágenes 2D.

  - *Occlusion*: permite generar partes no visibles en los elementos en RA, mediante objetos reales del entorno, para así conseguir un mayor realismo.

  - *Environment Probes:* permite generar un mapa cúbico de un área específica del entorno físico.

Esta versión de *ARCore Plugin utiliza* la versión de *ARCore 1.22 y* soporta las siguientes características: [24]

- *Device Localization*: localización del dispositivo.
- *Horizontal Plane Detection:* detección del plano horizontal.
- *Vertical Plane Detection:* detección del plano vertical.
- *Points Clouds:* es un conjunto de *feature points* y permiten un mapeo del espacio físico.
- *Pass-through Camera View:* permite analizar el entorno por el que pasa el campo de visión de la cámara.
- *Light Estimation:* estimación de la temperatura y brillo promedio del espacio.
- *Oriented Feature Points:* puntos característicos.
- *Hit Testing:* permite estimar impactos de *raycast*.
- *Session Management:* manejo del subsistema *session*.
- *ARCore Apk On-Demand Installation.*
- *2D Image Tracking:* reconocimiento y seguimiento de imágenes 2D.
- *Face Tracking:* reconocimiento y seguimiento de rostros.
- *Environment Probes:* permite generar un mapa cúbico de un área específica entorno físico.
- Occlusion: permite generar partes no visibles en los elementos en RA, mediante objetos reales del entorno, para así conseguir un mayor realismo.
- *Anchors*: permite fijar o anclar la posición de elementos, en RA, en el entorno.

Por otro lado, no soporta las siguientes características: [24], [36-39]

- *Object Tracking:* reconocimiento y seguimiento de objetos 3D.
- *Participant*: proporciona información sobre otros usuarios en una sesión de colaboración multiusuario.
- *Mesh:* reconstrucción 3D de entornos físicos mediante la generación eficiente de mallas.
- *Body Tracking*: interactuar con el cuerpo humano.

- **Editor Unity 2019.4.1f1**: la versión del editor de Unity 2019.4 la compatibilidad es correcta, es estable, no da fallos de compilación y muestra los elementos XR que proporciona el *SDK AR Foundation* al importarlo. [17], [23]

  Las pruebas realizadas, previamente, para comprobar que en esta versión funciona correctamente, han consistido en crear un marcador, y haciendo uso de un *script* capaz de reconocerlo, escanearlo y generar un *prefab GameObject Cube* en la pantalla del dispositivo.

- ***SketchUp Free**:* es un *software* que permite modelar en 3D con diferentes herramientas. [40]

  Se ha elegido este *software,* frente a otros, debido a que el tiempo de aprendizaje no es muy elevado, ofrece muchos recursos y es gratuito. Además, permite exportar objetos, que se pueden convertir en otras extensiones para importarlos en Unity.

- Herramienta ***arcoreimg***: es una herramienta, en línea, de comandos, que proporciona Google, y que se puede encontrar en el siguiente enlace. [41] Un diseño obtendrá un valor entre 0 y 100, pero para que sea óptima deberá obtener una puntuación de al menos 75 con el siguiente comando: "arcoreimg.exe eval-img --input_image_path=<<imagen>>", si se ejecuta en *Windows*, o "./arcoreimg eval-img --input_image_path=<<imagen>>", si se ejecuta en *Mac OS.* [42]

  Los marcadores, se crearán con diseños simples, eso sí, deberán ser lo suficientemente diferentes para que el *SDK* usado pueda identificarlos sin ningún tipo de problema y no llegue a confusión a la hora de generar los modelos 3D.

  En relación con el escaneo de marcadores, para realizar una lectura correcta, y evitar posibles errores, habrá que seguir el siguiente procedimiento: [43]

  - Evitar, en la medida de lo posible, enfocar a otro marcador diferente de manera muy rápida, sobre todo si este último tiene un tamaño y posición muy similar al anterior.
  - Llenar al menos el 25% el encuadre de la cámara.
  - El marcador deberá estar en buenas condiciones y sea plano, es decir, que no presente ningún desperfecto y no tenga curvatura.
  - La iluminación tiene que ser adecuada. Por tanto, se deberá evitar un entorno con poca o mucha luminosidad, ya que, en ambos casos, el marcador podría no reconocerse. Si la luminosidad fuese baja, este podría llegar a no distinguirse en el entorno, y en el caso contrario, podrían aparecer reflejos en el diseño.
  - El ángulo de enfoque, de la cámara, deberá ser adecuado.
  - Evitar movimientos rápidos de cámara y marcador.

A la hora de crear los diseños de los marcadores, se deberán cumplir las siguientes características, para que el escaneo y el seguimiento sea el deseado: [43]

- Resolución de al menos 300 x 300 píxeles.
- Formato *PNG* o *JPG.*
- No se utiliza información de color, es decir, una imagen en color, y la equivalente en escala de grises tendrán el mismo el resultado a la hora de ser escaneada.
- Evitar mucha compresión en las imágenes.
- Evitar códigos QR, códigos de barras, imágenes lineales, logotipos, etc.
- Evitar imágenes con patrones repetidos.

2.6 Diseño e interacción con el usuario

Este diseño quedará sujeto a cambios y es solo una versión provisional de lo que podría ser.

A continuación, se muestra una figura que representa la pantalla inicial de la aplicación:

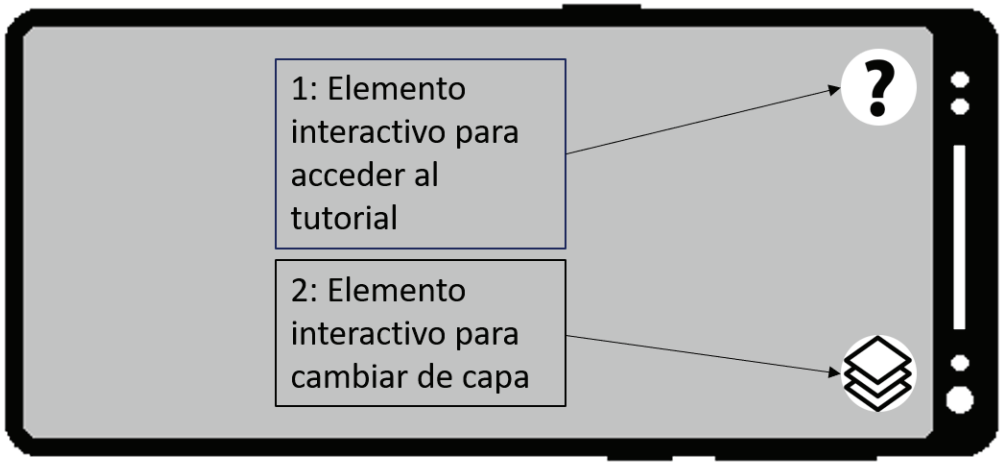

Figura 10 Prototipo de diseño de pantalla inicial de la aplicación. Elaboración propia (2020).

El usuario al iniciar la aplicación Unity, en el dispositivo, encontrará una interfaz con dos elementos interactivos, es decir, dos botones. Si pulsa el botón "1" (interrogación), mostrará una interfaz sencilla con una serie de instrucciones de cómo funciona la aplicación y qué pasos seguir. El diseño de esta interfaz se puede ver a continuación:

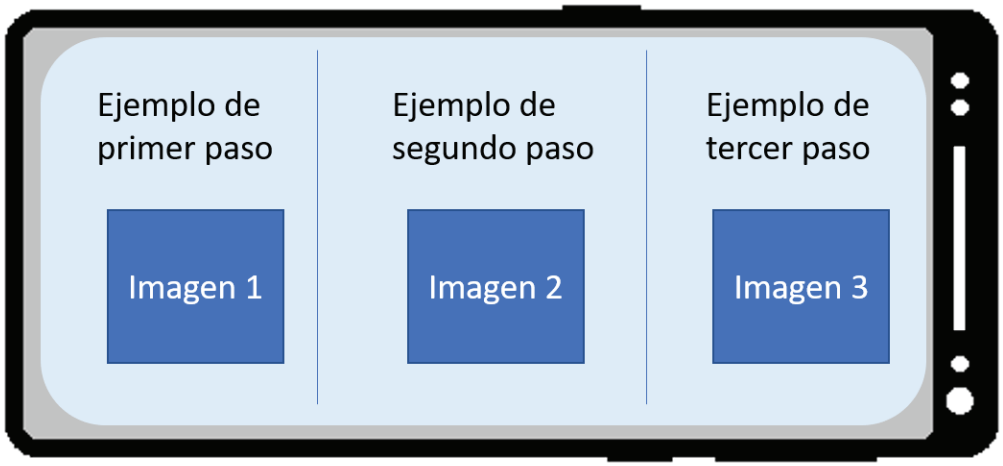

Figura 11 Prototipo de diseño de interfaz del tutorial. Elaboración propia (2020).

Cuando el usuario apunte y escanee el marcador (los marcadores, de elaboración propia, tendrán un diseño específico 2D, en blanco y negro, y que se proporcionarán para cada modelo. Además, lo recomendable será imprimirlos en papel, aunque también deberá poder ser reconocido en cualquier pantalla electrónica, ya que la cámara del dispositivo deberá ser capaz captar el diseño), se generará el modelo 3D, visualizándose en la pantalla del dispositivo.

Véase en las siguientes figuras un ejemplo de escaneo del marcador y generación de un ejemplo de modelo 3D:

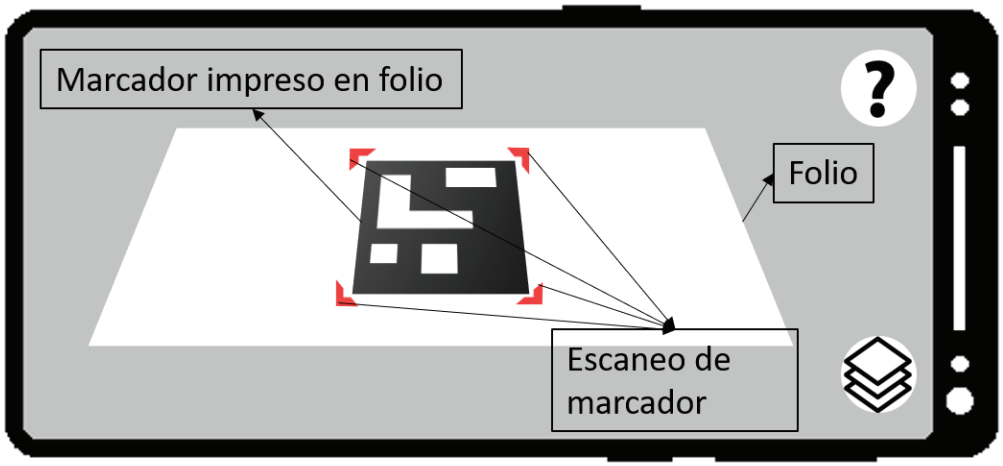

Figura 12 Prototipo de diseño de escaneo de un ejemplo de marcador. Elaboración propia (2020).

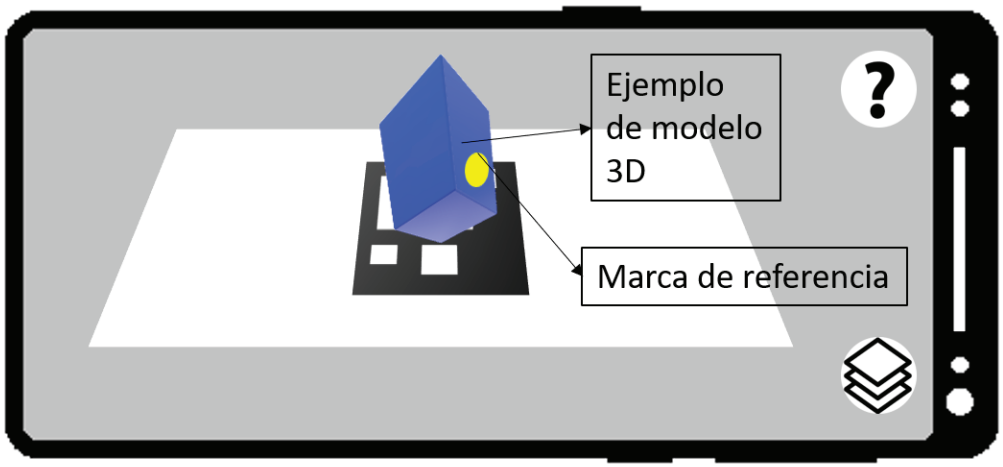

Figura 13 Prototipo de diseño de generación de ejemplo de modelo 3D. Elaboración propia (2020).

En este punto, si el usuario rota el marcador, el modelo deberá rotar también, ya que estará anclado a dicho elemento físico.

En la siguiente figura se puede observar cómo realizando un giro del marcador en sentido horario, el modelo 3D rota también.

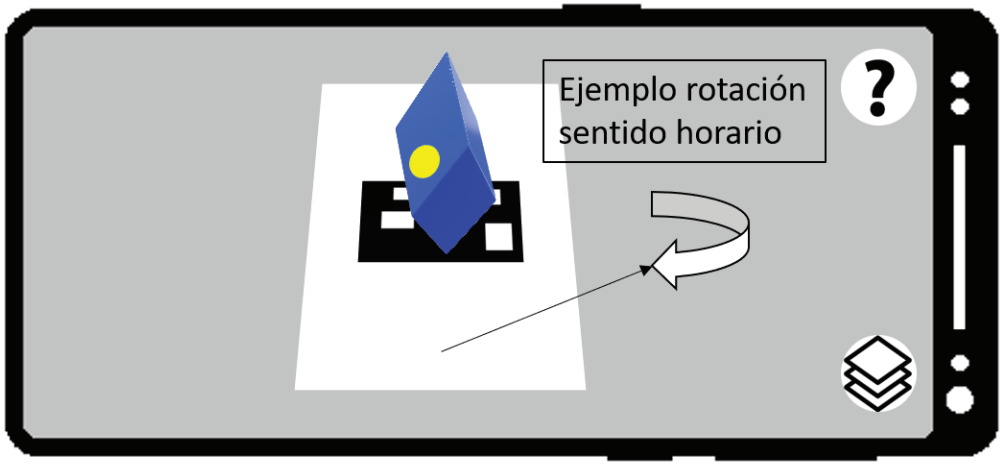

Figura 14 Ejemplo de visualización de ejemplo de modelo 3D al rotar el marcador. Elaboración propia (2020).

Si el usuario pulsa el botón de selección de capas, para ver los pisos del edificio, este comenzará a mostrar desde la última planta hasta la planta baja (efecto de destapar).

Así mismo, cuando se vaya mostrando cada capa, la información de los departamentos, aulas…, de cada piso, se mostrará simplemente información fijada al modelo, sin ningún tipo de interacción.

Véase en las figuras, que se muestran, un ejemplo de cómo funcionaría:

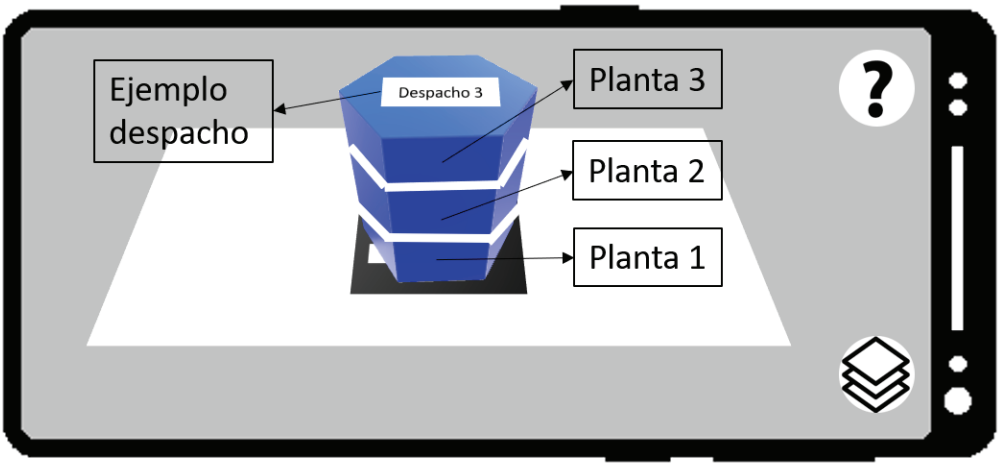

Figura 15 Prototipo de diseño de generación de modelo de ejemplo de edificio 1. Elaboración propia (2020).

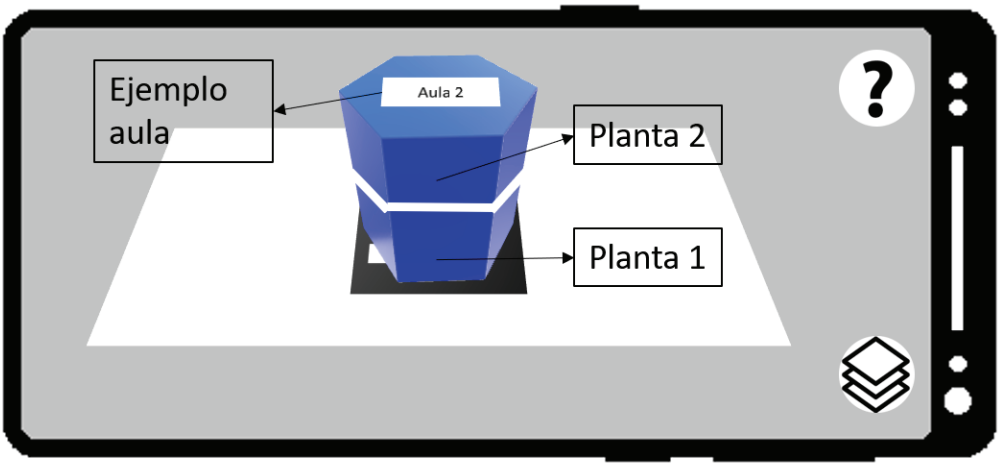

Figura 16 Ejemplo de selección de capas de modelo de ejemplo de edificio 1. Elaboración propia (2020).

Si el usuario quiere cambiar de edificio deberá enfocar a otro marcador.

Se puede observar en las siguientes figuras un ejemplo de un nuevo marcador y edificio:

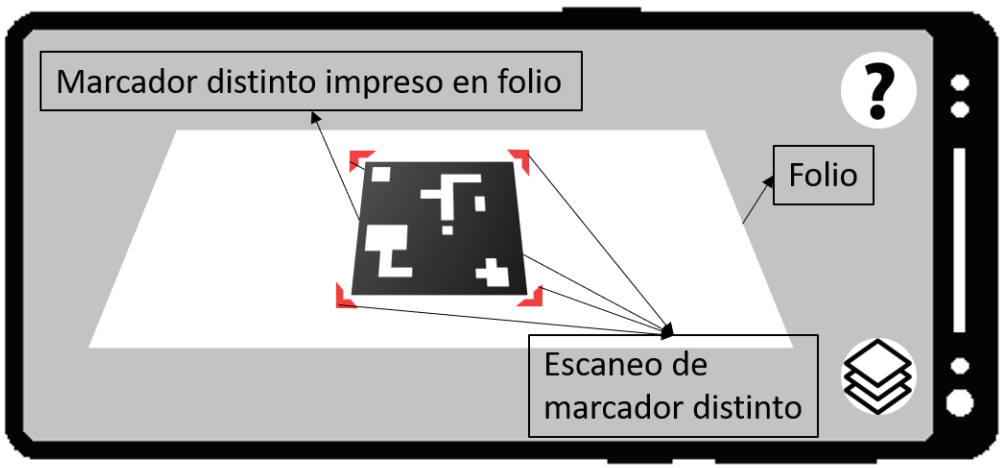

Figura 17 Prototipo de diseño de escaneo de un ejemplo de marcador 2. Elaboración propia (2020).

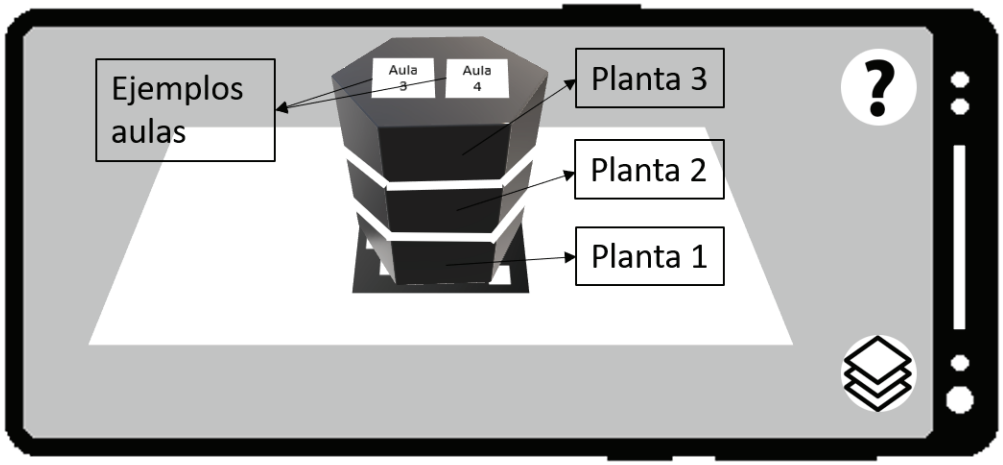

Figura 18 Prototipo de diseño de generación de modelo de ejemplo de edificio 2. Elaboración propia (2020).

2.7 Información previa acerca del desarrollo en Unity

A la hora de desarrollar las funcionalidades que se quieren alcanzar en este proyecto, se debe tener en cuenta que la arquitectura de Unity está basada en *scripts*. Por lo que, a la hora de implementar la solución de estos objetivos, habrá que ceñirse a los marcos que dicta esta plataforma y tener en cuenta que algunos diseños vienen impuestos.

Los *scripts* que maneja Unity están basados en los lenguajes *C#* o *UnityScript,* pero en este proyecto se utilizará únicamente el primero. Todo *script* que posee Unity, para su correcto funcionamiento, deriva de la clase *Monobehaviour,* la cual es la clase base, por lo que los *scripts* que se desarrollen también deberán derivar de esta. [44]

Por otro lado, en relación con los *scripts,* la clase *Monobehaviour* proporciona multitud de métodos, los cuales pueden sobrescribirse para proporcionar la funcionalidad deseada e implementarse cuando sea necesario. Algunos de los métodos son:

- *Awake():* es invocado cuando la instancia del *script* ha sido inicializada, se produce antes que *Start()*. [45]
- *Start():* es invocado justo cuando el *script* se habilita y antes de que se produzca alguna actualización en algún método. [46]
- *Update():* es invocado en cada fotograma. [47]
- *FixedUpdate():* es invocado cada cierto tiempo dependiendo de la configuración. *[48]*
- *OnDisable()*: es invocado cuando el *script* se deshabilita. [49]
- *OnEnable():* es invocado cuando el *script* se habilita y se activa. [50]

Estos son algunos de los métodos, para obtener información del resto véase el siguiente enlace. [44]

2.8 Análisis, diseño e implementación de la APP

A partir de este apartado, las funcionalidades, anteriormente descritas, en el apartado 5 del capítulo 2, se denominarán problemas.

2.8.1 Análisis del problema

Los diferentes problemas, que se van a abordar, se pueden clasificar en tres grupos:

- **Dependiente de *SDK AR Foundation* y *ARCore XR Plugin*:**
    - Escaneo y reconocimiento correcto del marcador, con diseño específico para cada modelo.
- **Modelado 3D:**
    - Generación del modelo 3D tras el escaneo.
    - Mostrar identificador (nombre o número) del aula, despacho, etc.
- **Investigación, manejo de Unity y desarrollo de soluciones para los problemas que se quieren abordar:**
    - Visualización correcta del modelo en 3D.
    - Selección de capas.
    - Pequeña guía de uso o tutorial.

2.8.2 Diseño de la solución

Una vez definidas las categorías, en la sección anterior, se puede dar paso a explicar cómo será una idea general de lo necesario para resolver los problemas expuestos.

- En relación con la primera categoría de problemas, será necesario la lectura de la guía o tutorial que proporciona Google, en su página oficial sobre desarrollo. Para poder lograr que el escaneo del marcador (el cual se creará con un diseño sencillo) y el reconocimiento de este sea el correcto, se usarán las características, atributos y funcionalidades que proporciona el *SDK AR Foundation*, acompañado de *ARCore XR Plugin*. Además, será necesario el desarrollo de varios *scripts* en *C#*, que se encargarán de poner en funcionamiento estas características, que proporciona este SDK. Estos *scripts* irán asociados a objetos que será necesario implementar en la ventana de jerarquía o *Hierarchy* de Unity.
- Por otro lado, para la segunda categoría será necesario la elección del software y posteriormente el estudio, en profundidad, de cómo funciona y de las características que ofrece para modelar, en 3D, las plantas de los edificios. Para ello, será imprescindible la lectura del manual, si lo hubiese, y realizar algún pequeño tutorial, para un rápido aprendizaje, y así poder obtener las nociones básicas para poder afrontar esta parte del proyecto.

Para abordar este problema, será imprescindible fotografiar por fuera y por dentro los bloques de la ETSIINF. De esta manera, se podrán tener las referencias necesarias, que permitirán el desarrollo de un modelado 3D que se asemeje lo máximo posible a la realidad. Se seguirá una escala fija, y un estándar común para cualquier parte o pieza del modelo. Se introducirá información fija en el modelo, que indique el número de despacho, de aula, planta, etc. Una vez desarrollado el diseño 3D, se exportará como un tipo de archivo determinado y se importará en Unity para poder usarlo.

- Por último, para la tercera categoría será necesario tener conocimientos sobre Unity y *C#* para poder abordarla.

  Para afrontar la correcta visualización 3D, se deberá interactuar con las proporciones del diseño, debido a que Unity no tiene una unidad de medida extrapolable al mundo reales decir, trabaja con sus propias unidades. En base a esto, habrá que mantener las proporciones entre los diferentes elementos, para que todo mantenga la relación de dimensiones con la realidad que se modela. Para ello, se tendrá que hacer pruebas hasta que el modelo se vea con un tamaño adecuado en la pantalla del dispositivo.

  En relación con la selección de capas, se deberá desarrollar una serie de *scripts* en *C#*, asociados a un objeto con el que el usuario pueda interactuar desde la pantalla del dispositivo, por ejemplo, un botón o pulsando sobre el propio modelo 3D; y que permita el correcto funcionamiento, por lo que se deberá estudiar las posibles soluciones para cumplir el objetivo. Por tanto, una de las ideas sería, asociar todos los modelos de las plantas de un bloque a un marcador determinado, cuando se escanee generar todas las capas del modelo, pero solamente una sea visible para el usuario, mientras que el resto no se puedan ver, hasta que se interactúe con el modelo, es decir, reciba un evento, y de esta forma llamar al *script*, que tendrá asociado, y que se encarga de cambiar de capa, es decir, desactivar la que estaba activa, y permitiendo la visibilidad en la siguiente.

  Por último, para afrontar cómo desarrollar la guía o el tutorial, será necesario saber qué información se incluirá, por ejemplo: texto, imágenes o vídeos. Una vez recopilada la información, investigar cómo generar interfaces en Unity, para poder plasmar dicho contenido, y desarrollar un *script* y asociarlo al botón, para que, al recibir un evento de interacción, con él, poder visualizar o cerrar el tutorial.

Ahora que ya se tiene una idea general de lo necesario, se dará paso a diseñar la solución, la cual se mostrará con imágenes en la sección de implementación.

Una vez que se tenga todo lo necesario importado, se crearán dos objetos *XR* en la jerarquía de Unity. Estos dos objetos, del mundo virtual, serán: *AR Session Origin* y *AR Session.* El primero de ellos gestionará el escaneo de imágenes y generación de los modelos 3D mediante componentes que proporciona el *SDK AR Foundation* y, además, contendrá un objeto llamado *AR Camera* (al cual habrá que añadirle un componente denominado *Physics Raycaster*, para gestionar los toques en la pantalla), el cual se encargará de gestionar la cámara del dispositivo. Por otro lado, el segundo de ellos se encargará de manejar la sesión de realidad aumentada, e inicialmente deberá permanecer deshabilitado.

En relación con los marcadores, se deberá crear un diseño para cada bloque, teniendo en cuenta las especificaciones que se han comentado en el punto 2.5.

Una vez creados, se deberán importar en Unity. El siguiente paso será crear un objeto *XR*, en el espacio destinado a la gestión de archivos, denominado *Reference Image Library,* al cual, se le adjuntarán los marcadores, y cada uno de ellos deberá tener un nombre específico. Este nombre deberá ser el mismo que el objeto que contenga el modelo 3D de cada bloque.

Una vez que se ha creado este objeto, se deberá añadir al objeto *AR Session Origin* el componente*,* que proporciona el *SDK,* llamado *AR Tracked Image Manager*, el cual posee un campo para insertar el objeto *Reference Image Library,* que se ha creado anteriormente. Este paso permitirá al *SDK* detectar, reconocer y extraer información de los marcadores que están adjuntados en el objeto mencionado con anterioridad.

Por otro lado, en relación con la creación de los modelos 3D, se hará uso del *software SketchUp Free*, por lo que será necesario tomar referencias de las ubicaciones de cada lugar de la ETSIINF mediante fotografías. Una vez que se hayan realizado, se realizará el modelo 3D de cada planta del bloque, y se exportará para, después, poder importarlo en Unity. Cuando se haya importado, será necesario que el modelo siga la estructura que se mostrará en la sección de implementación, debido a que los *scripts* que gestionen el modelado 3D, se basarán en esa estructura, por lo que, si no se cumple, se producirán errores. A estos modelos 3D, será necesario añadirles paneles con información de los lugares de interés de cada capa.

Dejando a un lado el modelado 3D, se pasará al diseño enfocado a los problemas relacionados con la correlación entre marcadores y modelos, la inicialización de toda la información para que pueda comenzar de manera correcta el proceso, la aparición, desaparición o seguimiento de estos, mientras se siga o se deje de escanear el marcador asociado a ellos; la posibilidad de cambiar de capa, mediante la interacción del usuario con el bloque o modelo y una correcta visualización de la información añadida a cada lugar de interés. Para ello, todos los problemas serán gestionados por el objeto, del mundo virtual, *AR Session Origin,* el cual poseerá unos componentes que estarán implicados en la funcionalidad y se mencionarán en la sección de implementación. A la hora de desarrollar todos estos problemas, se deberá tener en cuenta el punto 2.7.

Por último, en relación con lo que el usuario puede o no hacer, se hará alusión a la interfaz de la aplicación, por lo que podrá acceder a un tutorial, acceder a la realidad aumentada e interactuar con los modelos 3D. Para ello, la pantalla inicial se realizará mediante un objeto del mundo virtual de tipo *canvas*, al cual habrá que añadirle cierta información para que tenga la mayor usabilidad posible. Además, esta pantalla poseerá dos botones, uno para activar el tutorial, y otro para activar la realidad aumentada. Para ello, cada uno de estos botones se creará mediante un objeto, del mundo virtual, que ofrece Unity, denominado *Button* (el cual gestiona su propia estructura para añadirle la funcionalidad deseada). Haciendo alusión al botón del tutorial, si se pulsa, desactivará el *canvas* de la pantalla inicial y activará el del propio tutorial. El tutorial poseerá diferentes pantallas, las cuales mostrarán información textual y multimedia. Además, poseerá dos botones, uno para continuar y otro para retroceder (si se está en la primera pantalla y se retrocede, se activará el *canvas* de la pantalla inicial y se desactivará el del tutorial. De este mismo modo, si se está en la última pantalla y se avanza, se activará el de la pantalla inicial y se desactivará el del tutorial). En relación con el botón de la realidad aumentada, si se pulsa, se desactivará el *canvas* de la pantalla inicial y activará el objeto *AR Session,* el cual al iniciar la aplicación está desactivado.

A continuación, se muestra un sencillo diagrama de bloques que muestra el flujo de la aplicación:

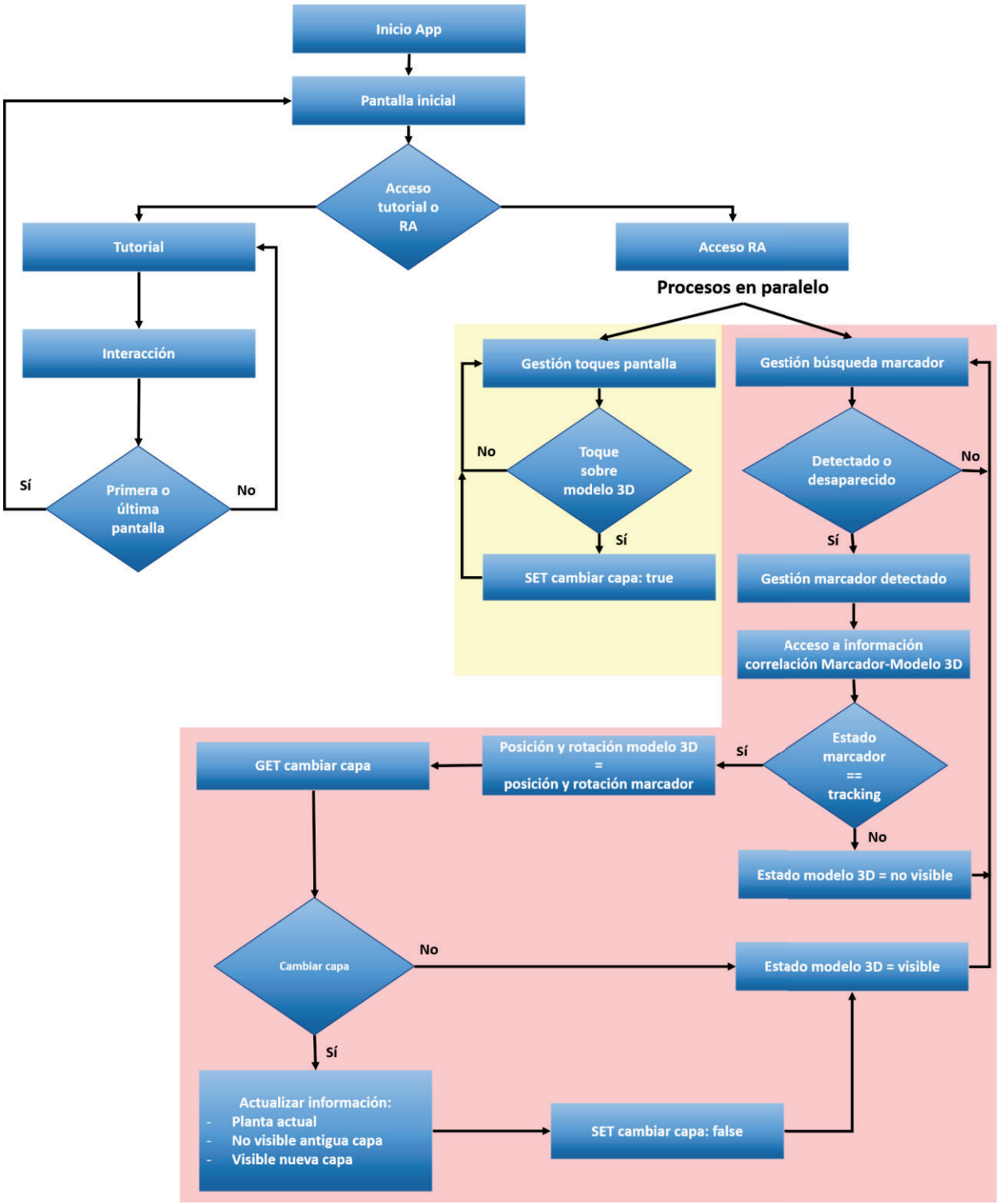

Figura 19 Diagrama de bloques del sistema, de la aplicación. Elaboración propia (2021).

2.8.3 Implementación

Esta sección estará basada en la anterior y mostrará, mediante imágenes, la estructura del entorno del proyecto y la implementación del diseño de la solución.

2.8.3.1 Preparación del entorno

Primero de todo, se deberá importar el *SDK AR Foundation* y *ARCore XR Plugin,* en Unity 2019.4.1f1, para así poder obtener los recursos, que proporcionan, y poder construir la aplicación basada en la realidad aumentada. Para ello, para poder importar el *SDK AR Foundation* y *ARCore XR Plugin,* habrá que seguir la siguiente ruta en Unity: "Window > Package Manager > All Packages". A continuación, se puede observar en una imagen:

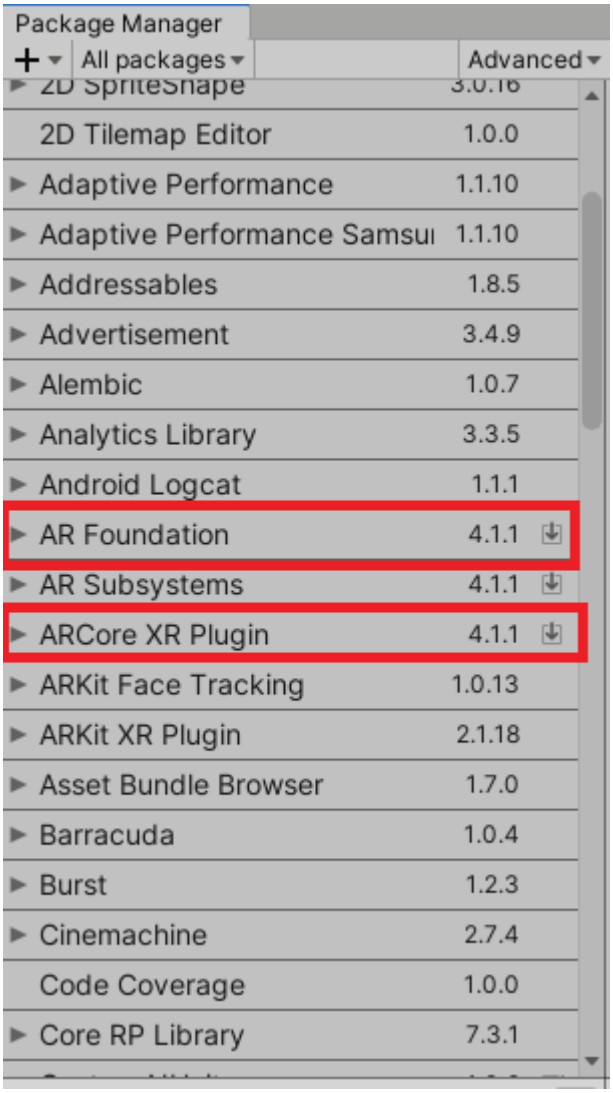

Figura 20 Ejemplo de importación de SDK y paquete. Elaboración propia (2021).

2.8.3.2 Creación de marcadores

Por otro lado, los marcadores que se han creado tendrán el siguiente diseño y se añadirá un ejemplo de uso de *arcoreimg* para verificar que son marcadores óptimos:

Figura 21 Diseño marcador para bloque 2. Elaboración propia (2021).

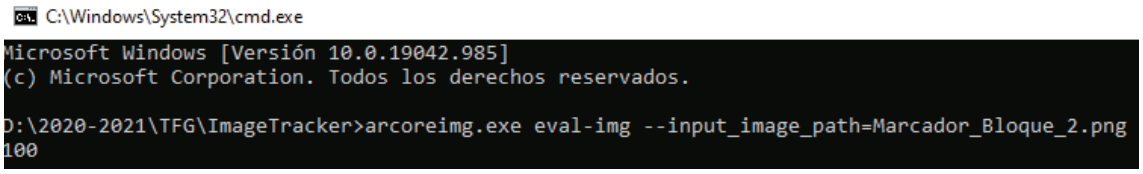

Figura 22 Verificar diseño óptimo del marcador 2, mediante arcoreimg. Elaboración propia (2021).

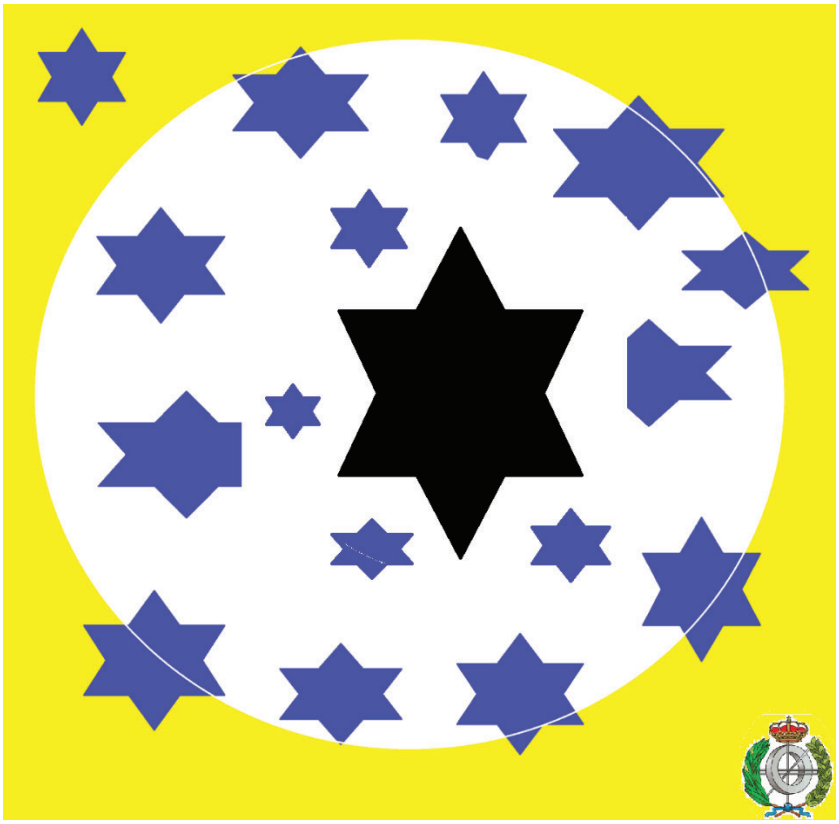

Figura 23 Diseño marcador para bloque 3. Elaboración propia (2021).

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19042.985]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\2020-2021\TFG\ImageTracker>arcoreimg.exe eval-img --input_image_path=Marcador_Bloque_3.png
100
```

Figura 24 Verificar diseño óptimo del marcador 3, mediante arcoreimg. Elaboración propia (2021).

2.8.3.3 Desarrollo en Unity

En relación con los objetos *AR Session Origin* y *AR Session,* se crearán en la jerarquía del entorno. Además, de forma automática se creará el objeto *AR Camera*. La jerarquía quedará de la siguiente manera:

Figura 25 Ejemplo creación AR Session Origin y AR Session. Elaboración propia (2021).

Debido a que al objeto *AR Session Origin* se le irán añadiendo componentes, se mostrará su estado final más adelante. Por lo que los objetos *AR Session* y *AR Camera* tendrán la siguiente estructura:

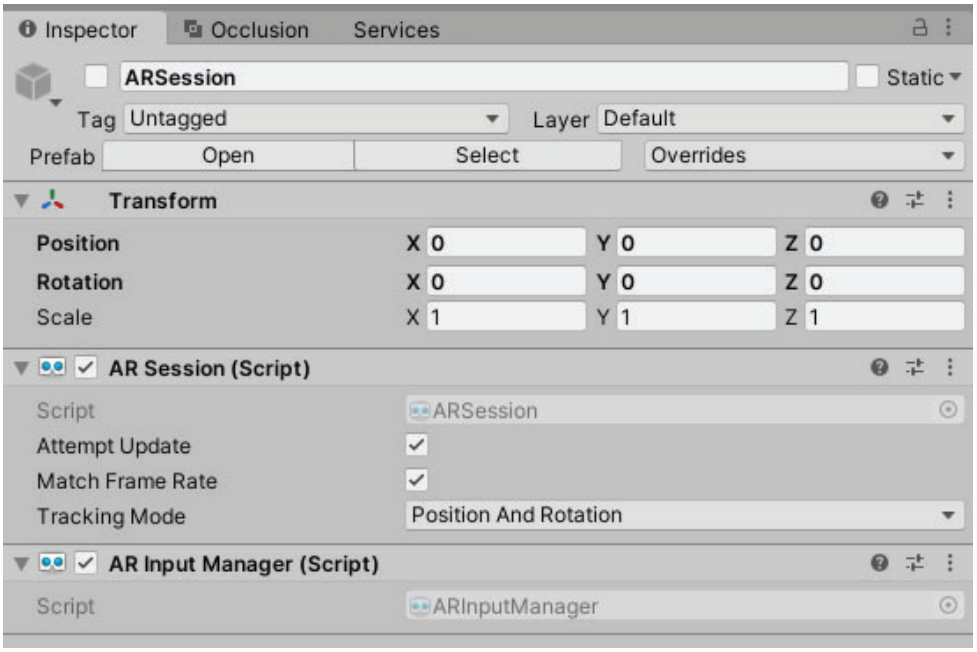

Figura 26 Ejemplo estructura objeto AR Session. Elaboración propia (2021).

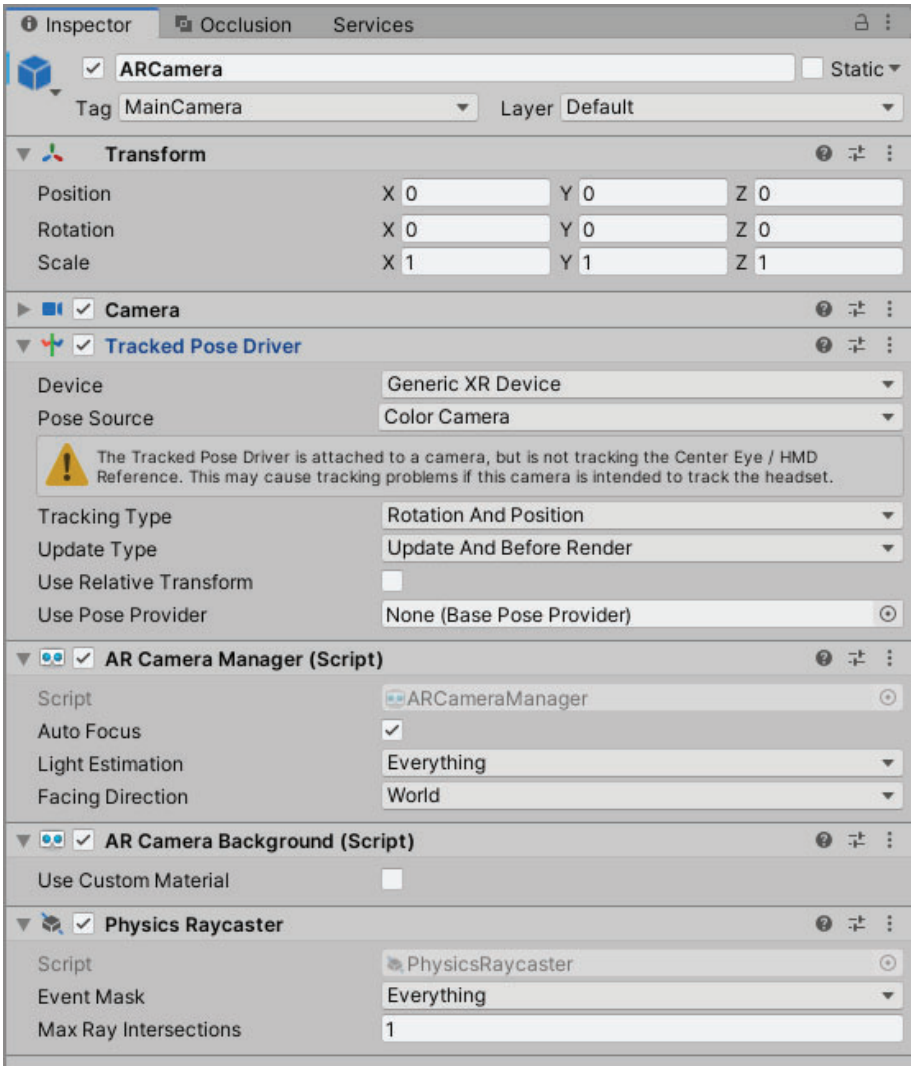

Figura 27 Estructura objeto AR Camera. Elaboración propia (2021).

Una vez que se han creado los marcadores, se creará el objeto *XR Reference Image Library* (con cualquier nombre) y se importarán los diseños en Unity. Véase en la siguiente imagen:

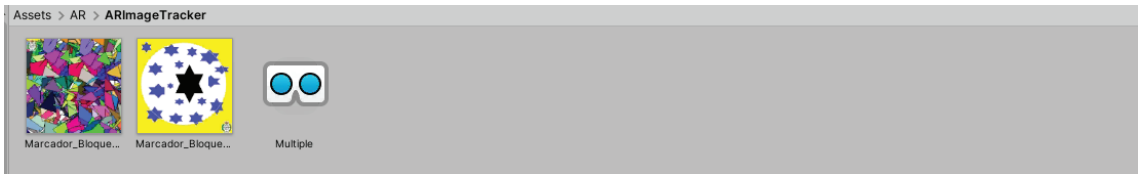

Figura 28 Marcadores y objeto XR Reference Image Library en Unity. Elaboración propia (2021).

Seguidamente, se adjuntarán los diseños al objeto denominado *Multiple*. Los nombres que se les asigna a los marcadores una vez adjuntados deben ser igual que el nombre del modelo 3D que representarán. En la siguiente imagen se muestra en detalle:

Figura 29 Marcadores adjuntados en el objeto XR Reference Image Library. Elaboración propia (2021).

Una vez adjuntados los marcadores, el objeto *XR Reference Image Library* se deberá insertar en el apartado *Serialized Library del script AR Tracked Image Manager* del objeto *AR Session Origin*. En la siguiente imagen se puede ver como quedaría la estructura, hasta ahora, de este objeto:

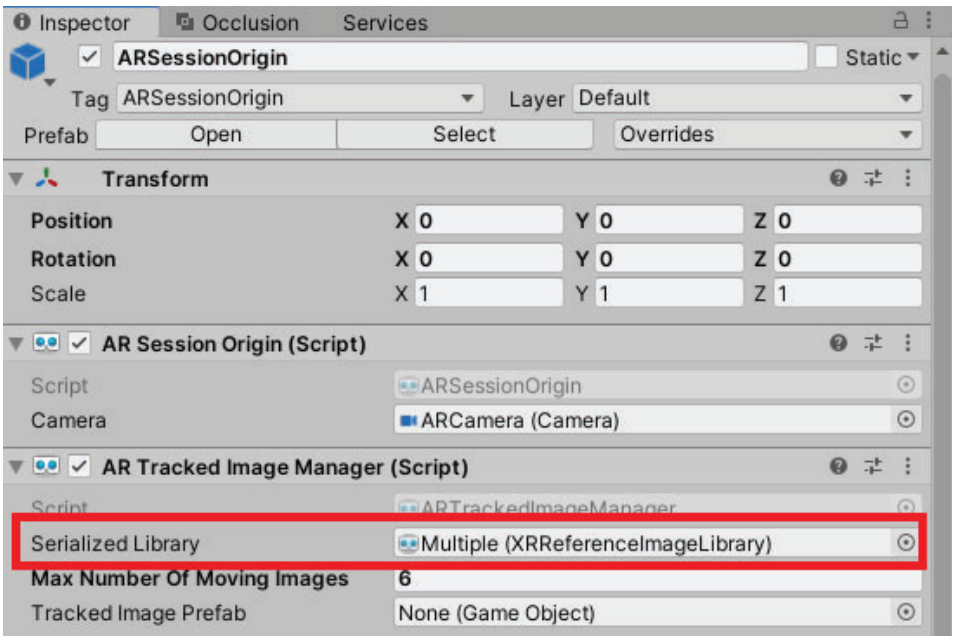

Figura 30 Implementado objeto XR Reference Image Library en en la variable Serialized Library del script AR Tracked Image Manager del objeto AR Session origin. Elaboración propia (2021).

En relación con los modelos 3D, se mostrarán imágenes de cómo son visualmente en el siguiente apartado, pero en este se importarán en Unity y tendrán la siguiente estructura:

Figura 31 Modelo 3D del bloque 2 importado en Unity. Elaboración propia (2021).

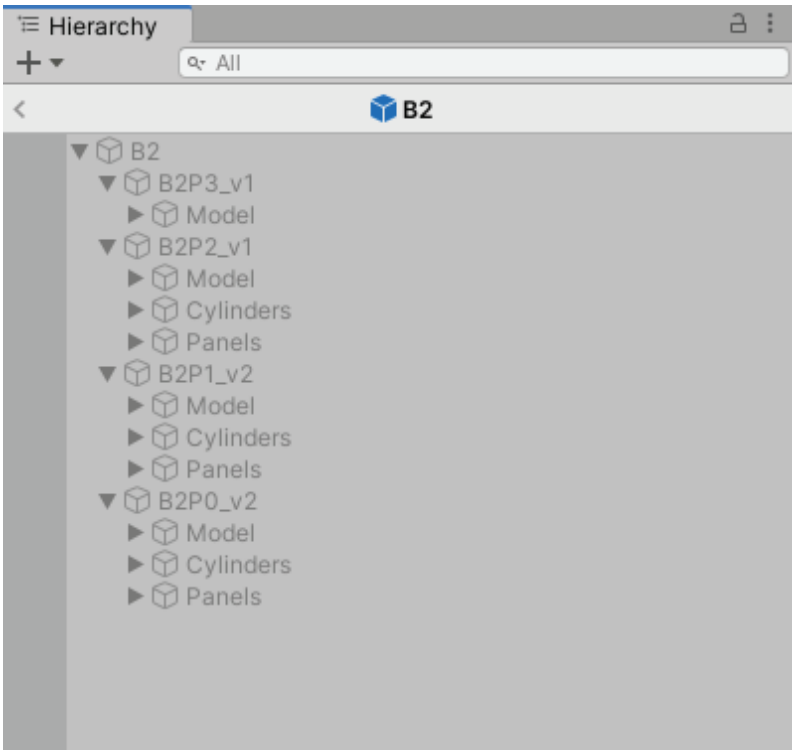

Figura 32 Estructura del modelo 3D del bloque 2. Elaboración propia (2021).

Figura 33 Modelo 3D del bloque 3 importado en Unity. Elaboración propia (2021).

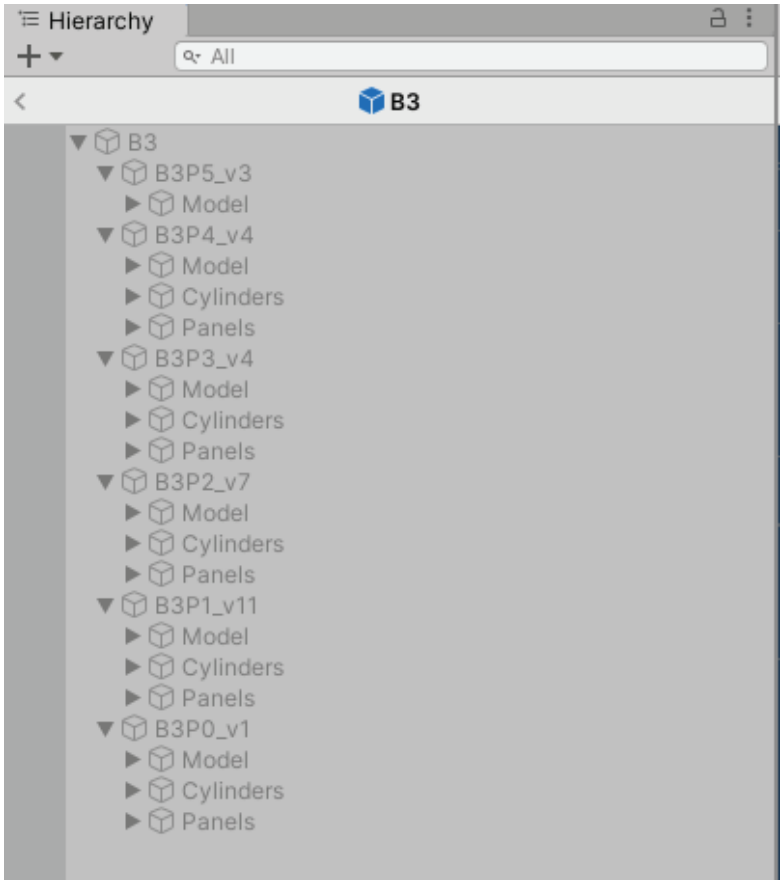

Figura 34 Estructura del modelo 3D del bloque 3. Elaboración propia (2021).

Como se puede observar, el primer hijo de cada bloque solo contiene a un objeto denominado *Model,* esto se debe a que al ser la última planta no necesita ni paneles (albergados en el objeto *Panels*) ni cilindros (albergados en el objeto *Cylinders.* Estos objetos son meramente decorativos para crear una animación), ya que en esta capa se muestra el bloque completo, sin lugares de interés.

Además, en Unity se importarán las texturas de los modelos 3D, véase en la siguiente figura:

Figura 35 Muestra de texturas usadas en los modelos 3D. Elaboración propia (2021).

Dejando a un lado el modelado 3D, se dará paso al *script* que los gestiona. Será un componente del objeto *AR Session Origin.* Para ello, se mostrarán imágenes del código utilizado. Este *script* trabajará junto al *script AR Tracked Image Manager,* mencionado anteriormente.

A continuación, se mostrará la declaración de las variables necesarias. Se puede observar la declaración de un diccionario, denominado *dataModel.* El cual tiene un campo "clave" de tipo *string* y un "valor" de tipo *Model.* La "clave" representa el nombre del modelo 3D, el cual coincide con el de su marcador asociado. El campo "valor" contiene un objeto *Model,* el cual contiene en su información: referencia a la instancia del objeto que contiene modelo 3D, un valor entero que representa la capa actual y un valor *bool* que indica si se desea cambiar de capa. Las siguientes figuras muestran la declaración de las variables y la clase *Model*:

```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;

[RequireComponent(typeof(ARTrackedImageManager))]
public class TrackedImageManager : MonoBehaviour
{
    // Objeto ARCamera.
    [SerializeField]
    private Camera arCamera;

    // Lista de los bloques que se van a ver en realidad aumentada.
    [SerializeField]
    private GameObject[] arObjectList;

    // Sonido al pulsar bloque.
    [SerializeField]
    private AudioSource touchSound;

    // Componente ARTrackedImageManager de objeto ARSessionOrigin.
    private ARTrackedImageManager m_TrackedImageManager;

    /* Diccionario que posee como clave el nombre del marcador (igual al del modelo 3D) y
       como valor un objeto Model, que contiene un GameObject que representa el modelo 3D,
       un int que indica la capa actual y un bool que indica si se ha solicitado cambiar de capa.
    */
    private Dictionary<string, Model> dataModel;
```

Figura 36 Declaración variables en script Tracked Image Manager. Elaboración propia (2021).

```
using UnityEngine;

public class Model : MonoBehaviour
{
    // Objeto modelo 3D.
    private GameObject arObject;

    // Capa actual.
    private int actualFloor;

    // Flag cambiar capa.
    private bool changeFloor;

    /**
     * Constructor:
     * - GameObject: Modelo 3D.
     * - int: Capa actual.
     * - bool: flag cambiar capa.
     */
    public Model(GameObject arObject, int actualFloor, bool changeFloor)
    {
        this.arObject = arObject;
        this.actualFloor = actualFloor;
        this.changeFloor = changeFloor;
    }

    /**
     * return modelo 3D.
     */
    public GameObject getARObject()
    {
        return this.arObject;
    }

    /**
     * return capa actual.
     */
    public int getActualFloor()
    {
        return this.actualFloor;
    }

    /**
     * return flag cambiar capa.
     */
    public bool getChangeFloor()
    {
        return this.changeFloor;
    }

    /**
     * Modifica capa actual.
     */
    public void setActualFloor(int actualFloor)
    {
        this.actualFloor = actualFloor;
    }

    /**
     * Modicia flag cambiar capa.
     */
    public void setChangeFloor(bool changeFloor)
    {
        this.changeFloor = changeFloor;
    }
}
```

Figura 37 Clase Model. Elaboración propia (2021).

Seguidamente, se muestra el método *Awake(),* el cual inicializa todas y cada una de las variables cuando se carga la instancia del *script*:

```
/**
 * Al despertar el proceso, inicializa todas las variales.
 */
void Awake()
{
    m_TrackedImageManager = GetComponent<ARTrackedImageManager>();

    dataModel = new Dictionary<string, Model>();

    foreach (GameObject arObject in arObjectList)
    {
        // Comprueba si la estructura del modelo 3D es correcto, en ese caso, pone el primer hijo a true,
        // para poder visualizarlo cuando se detecte su respectivo marcador.
        if (arObject.transform.childCount > 0)
            arObject.transform.GetChild(0).gameObject.SetActive(true);
        else throw new Exception("Estructura del modelo 3D incorrecto.");

        // Instancia los bloques con x = y = z = 0 y ninguna rotacion.
        GameObject newARObject = Instantiate(arObject, Vector3.zero, Quaternion.identity);

        /* Inicializa el diccionario con el nombre del modelo en el campo "clave" y un objeto Model
         en el campo clave, cuyo constructor requiere el modelo 3D, un entero que representa la planta actual
         y un bool que representa si se quiere cambiar de capa.
        */
        dataModel.Add(newARObject.name = arObject.name, new Model(newARObject, 0, false));
    }
}
```

Figura 38 Inicialización datos en script Tracked Image Manager. Elaboración propia (2021).

Por otro lado, el siguiente método es el que se encarga de la gestión de interacciones con la pantalla. Se puede observar la sentencia touchSound.*Play(),* la cual es un detalle que se ha añadido para que cuando se toque un modelo 3D se active un sonido. El método *Update()* estará constantemente controlando si se realizan interacciones con algún modelo 3D para cambiar de capa o planta, mediante toques en la pantalla. Véase la siguiente figura:

```
/**
 * Comprueba, en cada frame, si se pulsa la pantalla, en caso afirmativo,
 * se lanza un rayo y si colisiona con un bloque, activa el
 * flag de cambio de planta y genera el sonido "touchSound".
 */
private void Update()
{
    if (Input.touchCount > 0)
        foreach (Touch touch in Input.touches)
            if (touch.phase == TouchPhase.Began)
            {
                Ray ray = arCamera.ScreenPointToRay(touch.position);
                RaycastHit hit;

                if (Physics.Raycast(ray, out hit))
                {
                    dataModel[hit.collider.gameObject.name].setChangeFloor(true);
                    touchSound.Play();
                }

            }
}
```

Figura 39 Gestión interacción con pantalla y modelos 3D en script Tracked Image Manager. Elaboración propia (2021).

En la siguiente imagen se muestran los métodos *OnEnable()* y *OnDisable(),* los cuales serán invocados cuando un marcador sea detectado o haya desaparecido, gracias al *script AR Tracked Image Manager,* que es el que nos proporciona el *SDK AR Foundation* y gestiona esta funcionalidad.

```
/**
 * Detecta un marcador.
 */
void OnEnable()
{
    m_TrackedImageManager.trackedImagesChanged += OnTrackedImagesChanged;
}


/**
 * No detecta marcador
 */
void OnDisable()
{
    m_TrackedImageManager.trackedImagesChanged -= OnTrackedImagesChanged;
}
```

Figura 40 Métodos llamados cuando se detecta un marcador en script Tracked Image Manager. Elaboración propia (2021).

A continuación se presenta el método intermediario entre los anteriores y el método *UpdateARObject(*ARTrackedImage *trackedImage)*:

```
/**
 * Gestiona la detección de los marcadores:
 * - Marcador nuevo.
 * - Posición de marcador ya reconocido previamente.
 */
void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    foreach (ARTrackedImage trackedImage in eventArgs.added)
        UpdateARObject(trackedImage);


    foreach (ARTrackedImage trackedImage in eventArgs.updated)
        UpdateARObject(trackedImage);


    foreach (ARTrackedImage trackedImage in eventArgs.removed)
        arObjects[trackedImage.referenceImage.name].SetActive(false);

}
```

Figura 41 Método intermediario entre detección y gestión de modelos 3D en script Tracked Image Manager. Elaboración propia (2021).

Por último, en relación con el *script,* se muestra el método que gestiona los modelos 3D. Este método es invocado por el método *OnTrackedImagesChanged (ARTrackedImagesChangedEventArgs eventArgs),* constantemente, mientras se esté detectando un marcador, debido a que la información del modelo 3D (como la posición), debe estar actualizado para conseguir un seguimiento fluido. La finalidad de este método es permitir que, cuando se detecte un marcador, se haga visible el modelo 3D que tiene asociado. De este mismo modo, cuando el marcador deje de detectarse, el modelo 3D, que tiene asociado, deje de ser visible. Por tanto, el método deberá extraer la información necesaria del marcador, buscar en el diccionario que modelo 3D le corresponde y comprobar si el estado del marcador es igual a *tracking.* Si el estado no es el mencionado, el modelo 3D que le corresponde al marcador deberá pasar a ser no visible, ya que el marcador habrá desaparecido. Por otro lado, si el estado sí es el

mencionado, se deberá actualizar la posición y tamaño del modelo, para así mantener un seguimiento correcto. Se comprobará si el *flag* para cambiar de capa está activo, en ese caso, se actualizará la información del diccionario *dataModel*. Por último, el estado del modelo 3D pasará a ser visible. Además, se puede observar una sentencia que se ha añadido como detalle, la cual permite que los paneles siempre estén orientados hacia la cámara. Véase la siguiente figura:

```
/**
 * Gestiona la aparición o desaparición de los bloques asociados al correspondiente marcador,
 * así como, el cambio de plantas de cada bloque.
 */
private void UpdateARObject(ARTrackedImage trackedImage)
{
    // Copia nombre.
    string referenceTrackedImageName = trackedImage.referenceImage.name;

    // Referencia objeto modelo 3D.
    GameObject arObject = dataModel[referenceTrackedImageName].getARObject();

    // Comprueba si el estado del marcador se está trackeando.
    // En caso contrario, se desactiva el bloque y desaparece de la cámara.
    if (trackedImage.trackingState == UnityEngine.XR.ARSubsystems.TrackingState.Tracking)
    {
        // Objeto adquiere misma posición y rotación que el marcador.
        arObject.transform.position = trackedImage.transform.position;
        arObject.transform.rotation = trackedImage.transform.rotation;

        // Tamaño del objeto pasa de 0 ,0, 0 a 1, 1, 1, para que se pueda visualizar.
        arObject.transform.localScale = new Vector3(1, 1, 1);

        // Número de planta actual.
        int i = dataModel[referenceTrackedImageName].getActualFloor();

        // Comprueba si el flag para cambiar de planta del bloque esta activado.
        if (dataModel[referenceTrackedImageName].getChangeFloor())
        {
            // Número de hijos.
            int childCount = arObject.transform.childCount - 1;

            // Comprueba si i es el ultimo hijo, en ese caso, pasa a estar activo el primer hijo e inactivo el
              último.
            // En caso contrario desactiva el actual hijo y activa el siguiente.
            if (i == childCount)
            {
                i = 0;
                arObject.transform.GetChild(childCount).gameObject.SetActive(false);
                arObject.transform.GetChild(i).gameObject.SetActive(true);
            }
            else
            {
                arObject.transform.GetChild(i).gameObject.SetActive(false);
                arObject.transform.GetChild(++i).gameObject.SetActive(true);
            }

            // Se actualiza en el diccionario la planta actual, del respectivo bloque.
            dataModel[referenceTrackedImageName].setActualFloor(i);

            // Se desactiva el flag para cambiar de planta.
            dataModel[referenceTrackedImageName].setChangeFloor(false);
        }

        // Visualización habilitada.
        if (!arObject.activeSelf)
            arObject.SetActive(true);

        // Comprueba si hay paneles en el objeto. Si hay, hace que miren en la dirección de la cámara.
        if (arObject.transform.GetChild(i).Find("Panels") != null)
        {
            Transform panels = arObject.transform.GetChild(i).Find("Panels");
            for (int j = 0; j < panels.childCount; j++)
                panels.GetChild(j).LookAt(transform.GetChild(0).transform.position);
        }
    }

    else
    {
        // Visualización deshabilitada.
        if (arObject.activeSelf)
            arObject.SetActive(false);
    }
}
```

Figura 42 Método gestión de modelos 3D en en script Tracked Image Manager. Elaboración propia (2021).

Añadiendo este *script* al objeto *AR Session Origin*, la estructura de este quedaría de la siguiente forma:

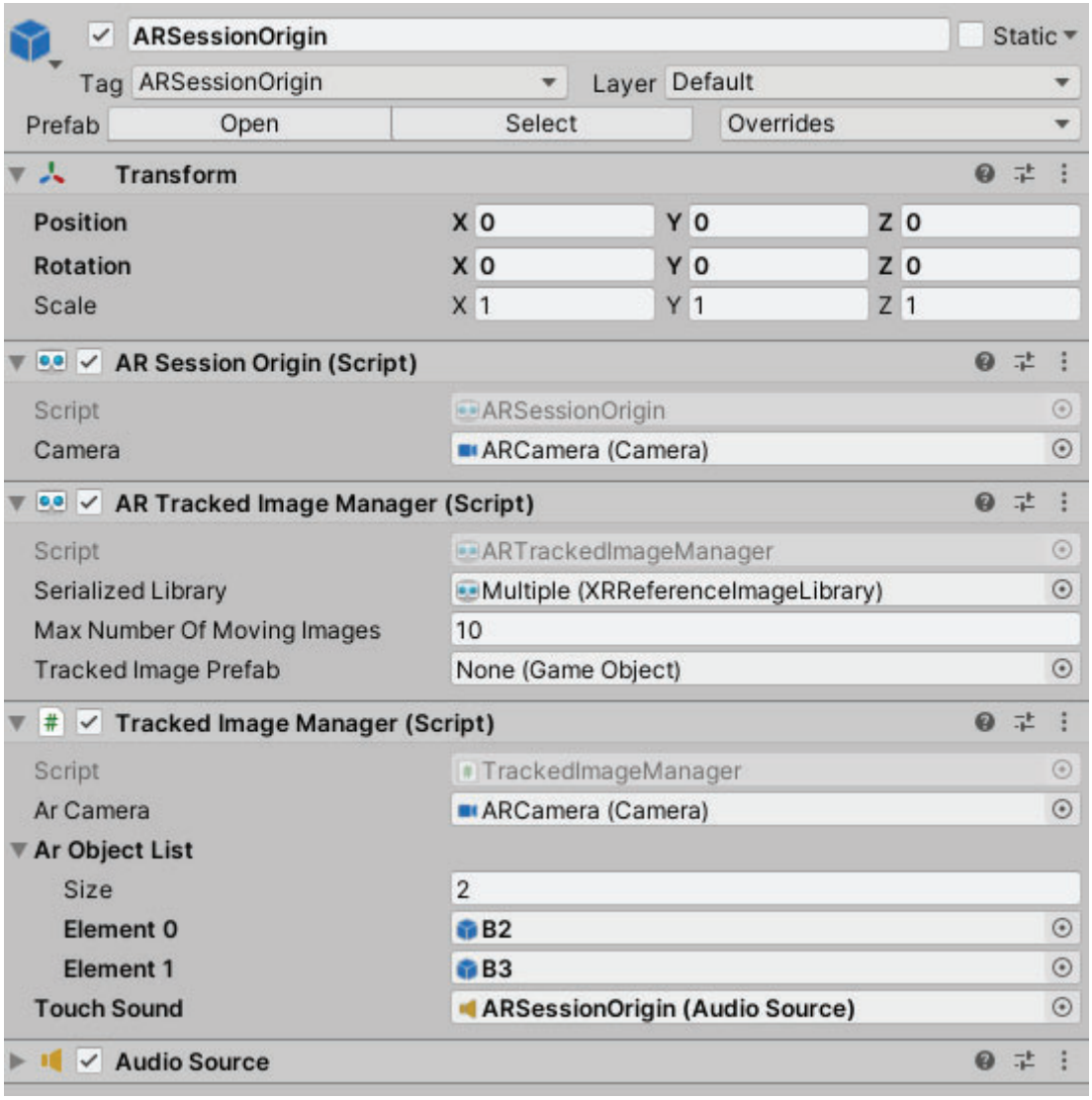

Figura 43 Estructura final objeto AR Session Origin. Elaboración propia (2021).

En relación con el *canvas* de la pantalla inicial, se puede observar cómo quedará el diseño en la siguiente imagen:

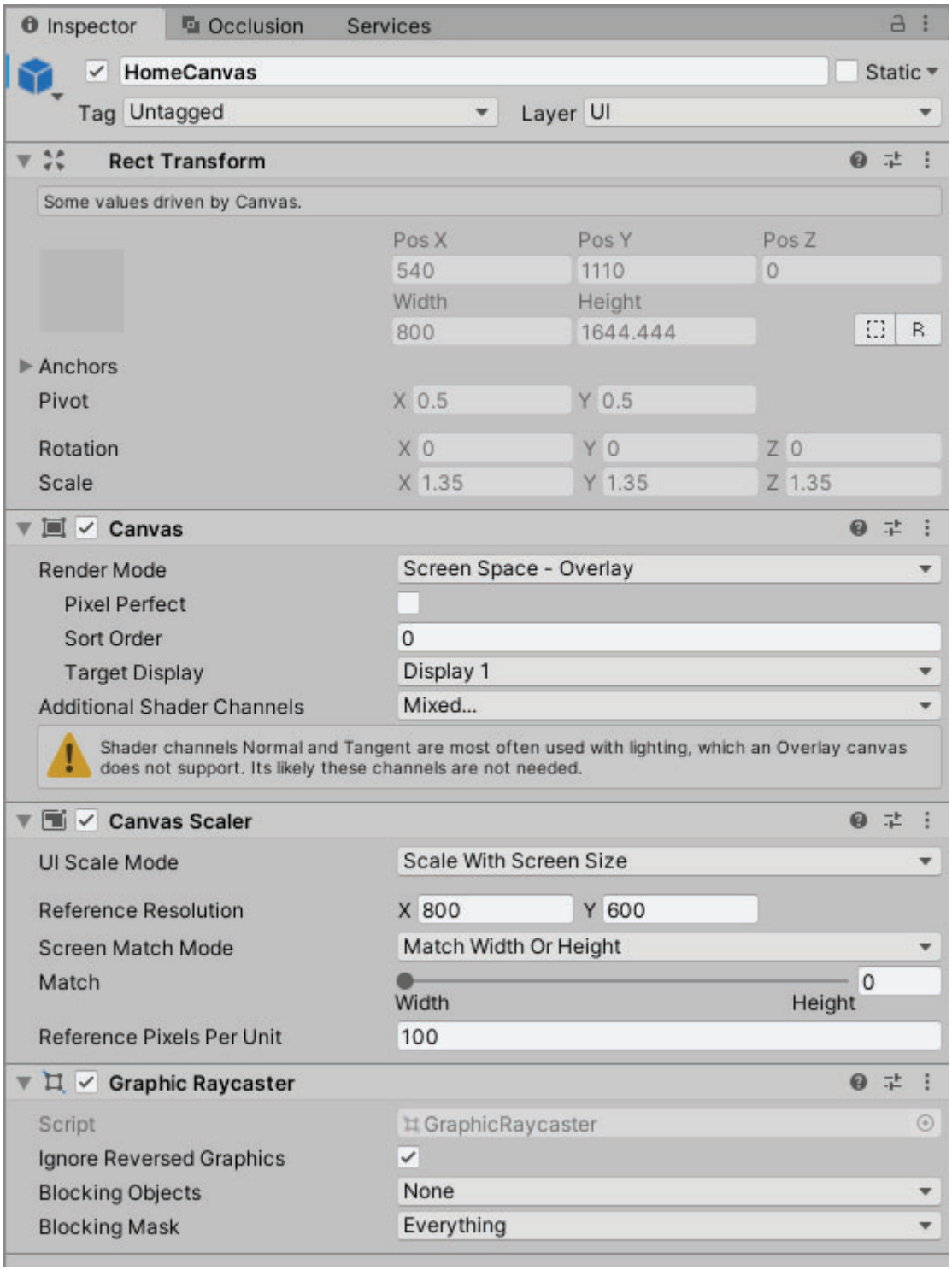

Figura 44 Estructura canvas pantalla inicial. Elaboración propia (2021).

Figura 45 Canvas pantalla inicial. Elaboración propia (2021).

Los botones que acceden a la parte *Tutorial* y a *RA Cámara*, serán gestionados mediante el siguiente *script.* Se puede observar que hay dos variables de tipo *GameObject,* las cuales contienen el valor del "hijo 0" y del "hijo 1" del objeto *canvas.* Véase la siguiente imagen:

```
using UnityEngine;

public class MainController : MonoBehaviour
{
    // Objeto Canvas.
    [SerializeField]
    private GameObject canvas;

    // Objeto ArSession.
    [SerializeField]
    private GameObject arSession;

    // Variables que dan nombre a los literales: hijo 0 y 1 del objeto Canvas.
    private GameObject tutorialCanvas, homeCanvas;

    /**
     *  Orienta la pantalla en modo retrato, de manera fija, en la pantalla principal.
     *  Añade valor a las variables tutorialCanvas y homeCanvas:
     *  - tutorialCanvas = hijo 0.
     *  - homeCanvas = hijo 1.
     */
    private void Awake()
    {
        Screen.orientation = ScreenOrientation.Portrait;
        tutorialCanvas = canvas.transform.GetChild(0).gameObject;
        homeCanvas = canvas.transform.GetChild(1).gameObject;
    }


    /**
     * Activa el tutorial.
     */
    public void EnableTutorial()
    {
        homeCanvas.SetActive(false);
        tutorialCanvas.SetActive(true);
    }


    /**
     * Activa la realidad aumentada.
     */
    public void EnableAR()
    {
        homeCanvas.SetActive(false);
        arSession.SetActive(true);
    }
}
```

Figura 46 Script gestión botones pantalla inicial. Elaboración propia (2021).

Por último, se mostrarán imágenes del tutorial, acerca de la estructura del *canvas*, en Unity, y cómo es su diseño visual, el cual posee información descriptiva, en forma de texto, e información adicional, en forma de imágenes o vídeos.

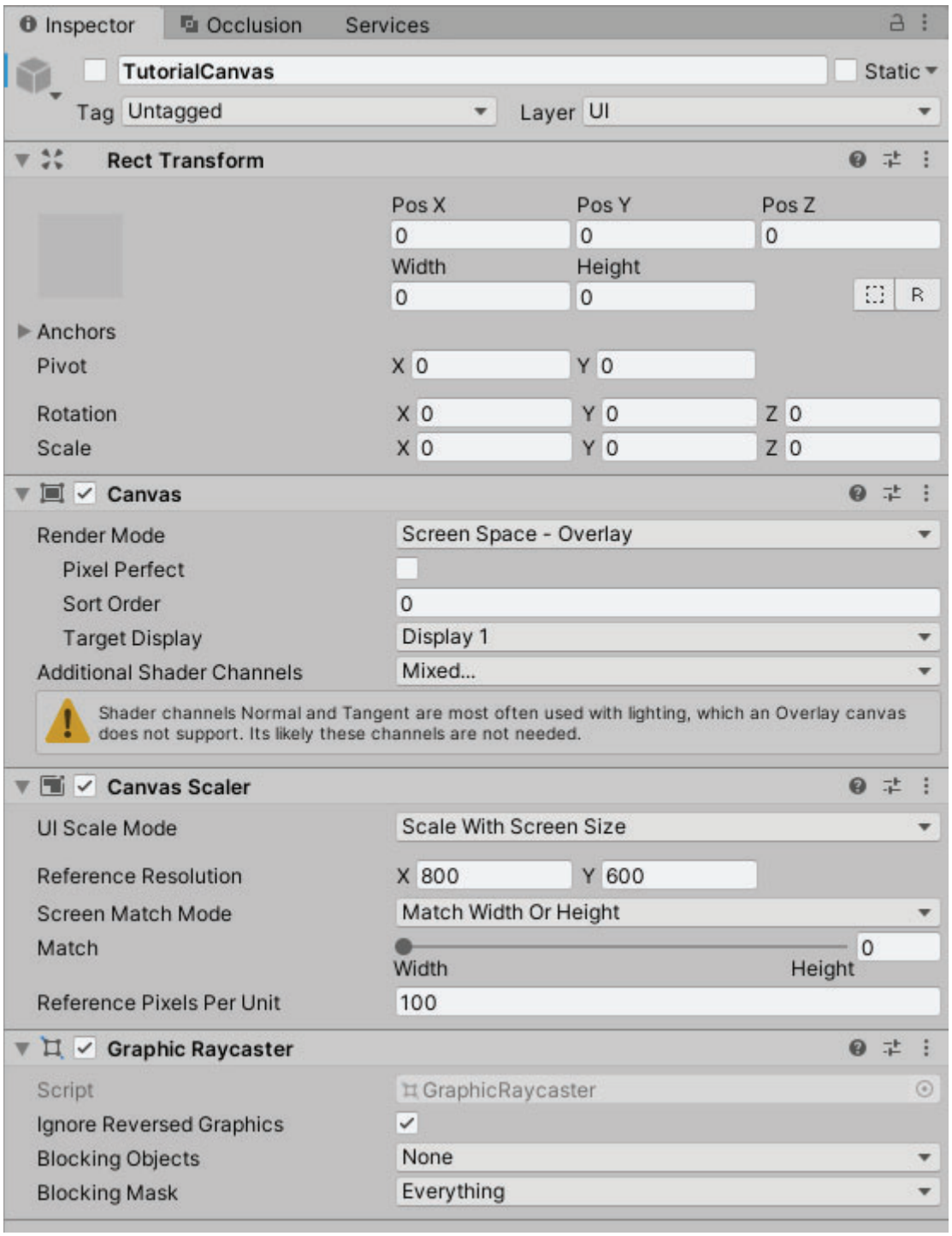

Figura 47 Estructura canvas tutorial. Elaboración propia (2021).

Figura 48 Primera pantalla canvas tutorial. Elaboración propia (2021).

Figura 49 Segunda pantalla canvas tutorial. Elaboración propia (2021).

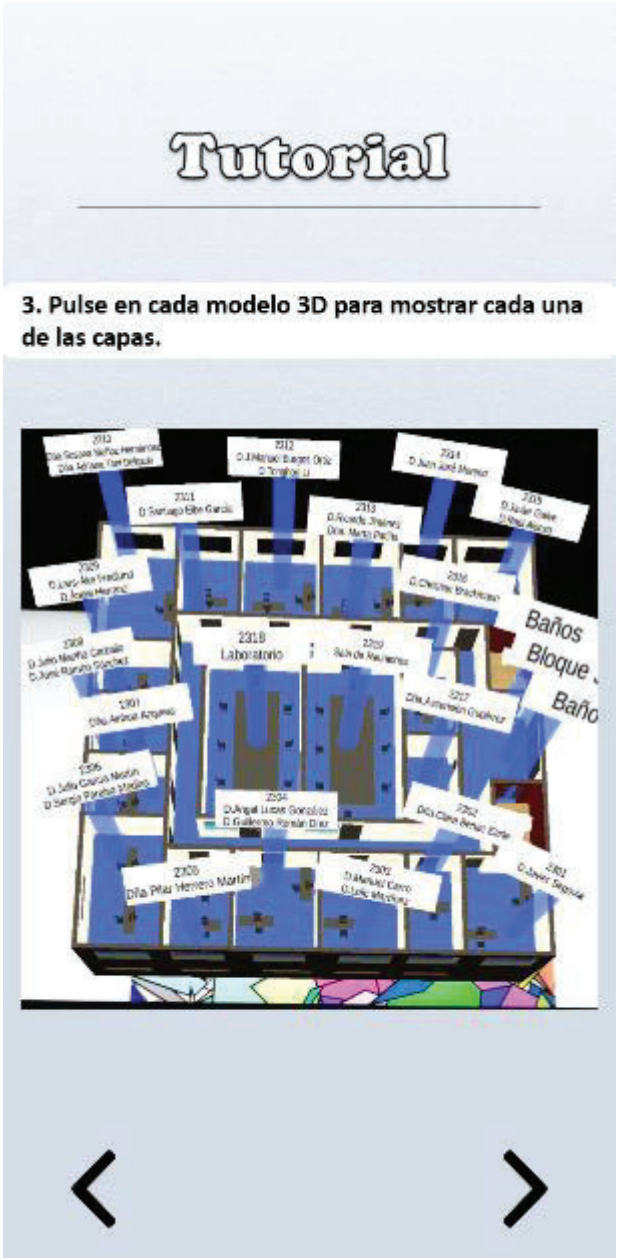

Figura 50 Tercera pantalla canvas tutorial. Elaboración propia (2021).

Figura 51 Cuarta pantalla canvas tutorial. Elaboración propia (2021).

Figura 52 Quinta pantalla canvas tutorial. Elaboración propia (2021).

La gestión de los botones del tutorial se realizará mediante el siguiente *script*, mostrado en la figura:

```
using UnityEngine;

public class TutorialController : MonoBehaviour
{
    // Objeto pantallas tutorial.
    [SerializeField]
    private GameObject steps;

    // Canvas pantalla inicio.
    [SerializeField]
    private GameObject homeCanvas;

    // Canvas pantalla tutorial.
    [SerializeField]
    private GameObject tutorialCanvas;

    // Variable estado actual.
    private static int actualStep;

    /**
     * Inicializa pantalla actual.
     */
    private void Awake()
    {
        actualStep = 0;
    }


    /**
     * Gestiona ir para la pantalla anterior del tutorial.
     */
    public void Back()
    {
        if (actualStep == 0)
        {
            homeCanvas.SetActive(true);
            tutorialCanvas.SetActive(false);
        }
        else
        {
            steps.transform.GetChild(actualStep--).gameObject.SetActive(false);
            steps.transform.GetChild(actualStep).gameObject.SetActive(true);
        }
    }


    /**
     * Gestiona ir para la pantalla siguiente del tutorial.
     */
    public void Next()
    {
        if (actualStep == steps.transform.childCount-1)
        {
            steps.transform.GetChild(actualStep).gameObject.SetActive(false);
            actualStep = 0;
            homeCanvas.SetActive(true);
            tutorialCanvas.SetActive(false);
            steps.transform.GetChild(actualStep).gameObject.SetActive(true);
        }
        else
        {
            steps.transform.GetChild(actualStep++).gameObject.SetActive(false);
            steps.transform.GetChild(actualStep).gameObject.SetActive(true);
        }
    }
}
```

Figura 53 Script gestión botones tutorial. Elaboración propia (2021).

La jerarquía final y la estructura del proyecto en Unity quedará de la siguiente forma:

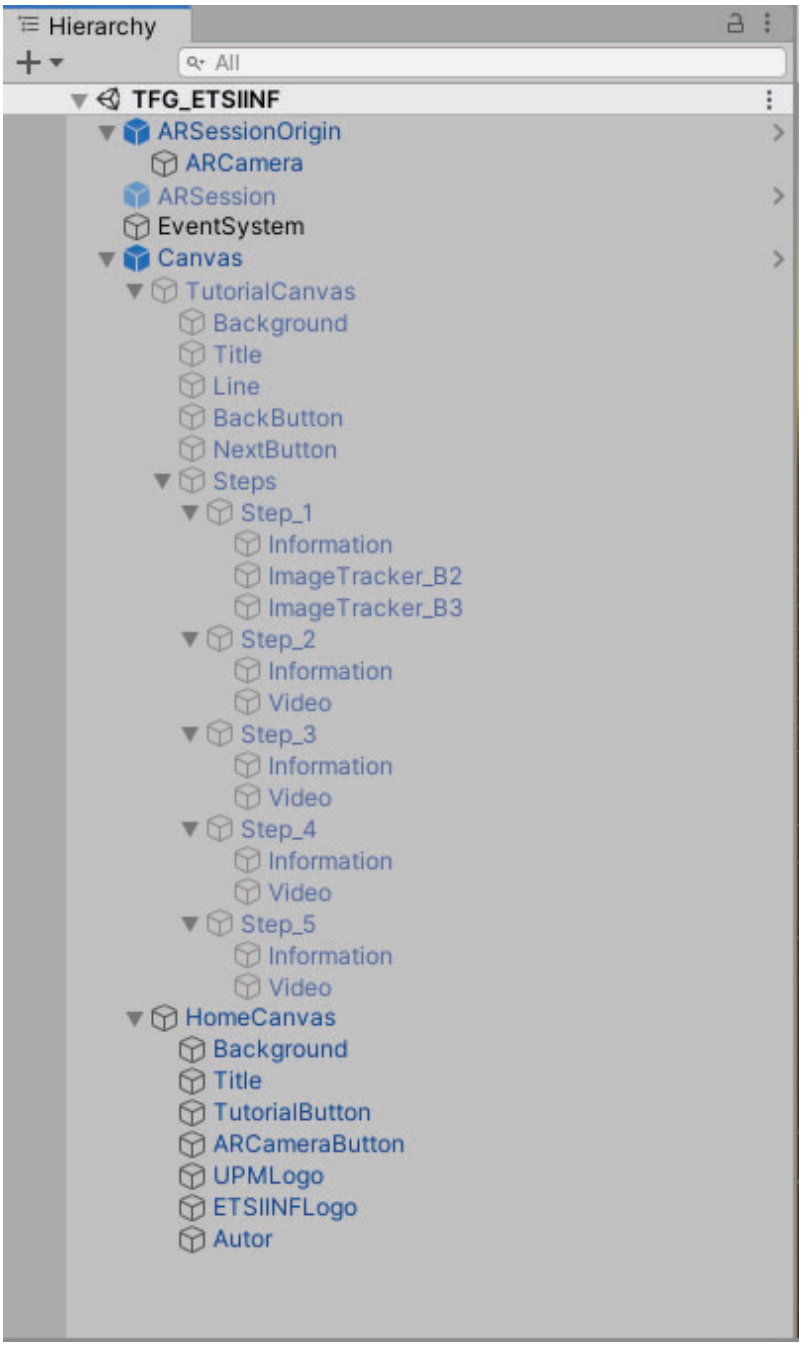

Figura 54 Estado final de la jerarquía del proyecto en Unity. Elaboración propia (2021).

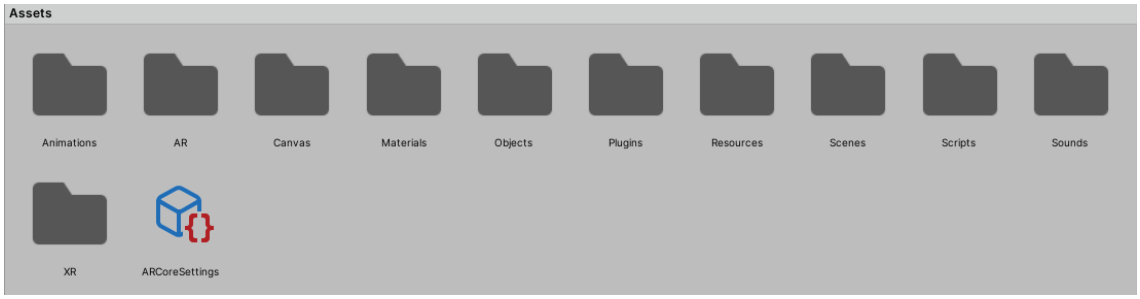

Figura 55 Estado final de la gestión de archivos del proyecto en Unity. Elaboración propia (2021).

Para terminar, y para conseguir un aspecto más dinámico, mediante el siguiente *script* se habilitarán animaciones para los modelos 3D de los bloques, cilindros y paneles. véase la siguiente figura:

```
using UnityEngine;

public class ActivateAnimation : MonoBehaviour
{
    // Objeto que tiene que aparecer.
    [SerializeField]
    private GameObject obj;

    // Tamaño al que tiene que llegar.
    [SerializeField]
    private double maxY;

    /**
     * Permite la aparición del bloque mediante una animación, actualizándose cada frame.
     */
    void Update()
    {
        gameObject.GetComponent<Animator>().enabled = obj.transform.localScale.y == maxY;
    }
}
```

Figura 56 Gestión animaciones bloques, cilindros y paneles. Elaboración propia (2021).

2.9 Construcción del modelo 3D

En este apartado se mostrarán los modelos 3D, desarrollados mediante *SketchUp Free,* de los bloques 2 y 3 de la ETSIINF.

2.9.1 Modelo 3D bloque 2

Esta sección contendrá las imágenes que representan el modelo de dicho bloque. Se puede observar en cada figura cada una de las plantas que será representada en la aplicación.

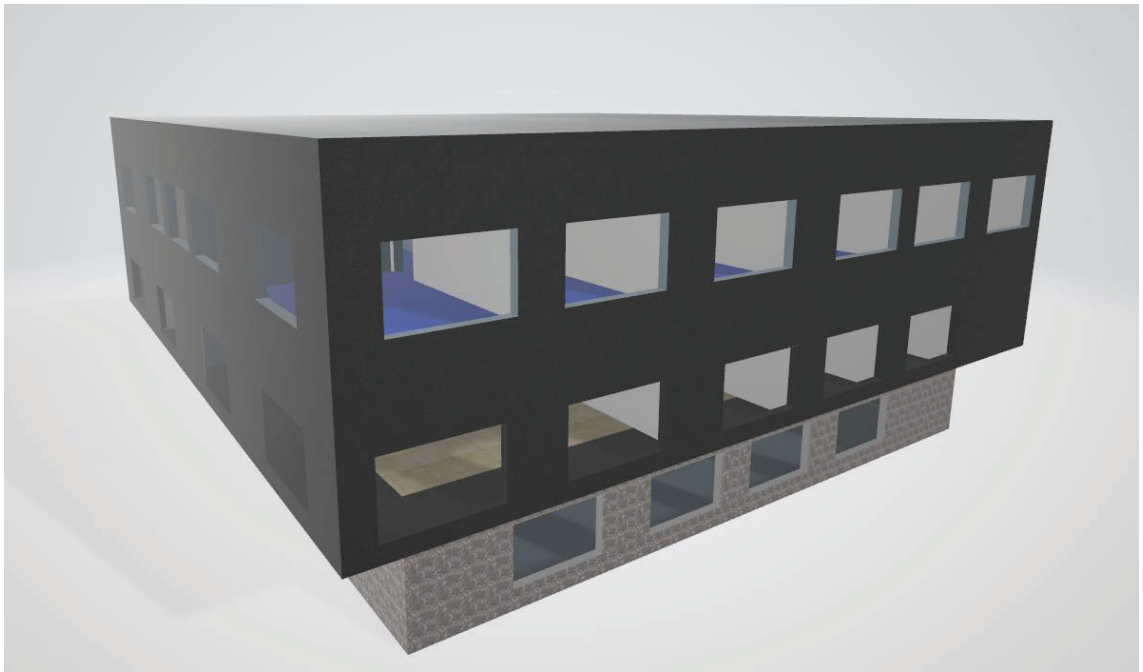

Figura 57 Bloque 2 completo. Elaboración propia (2021).

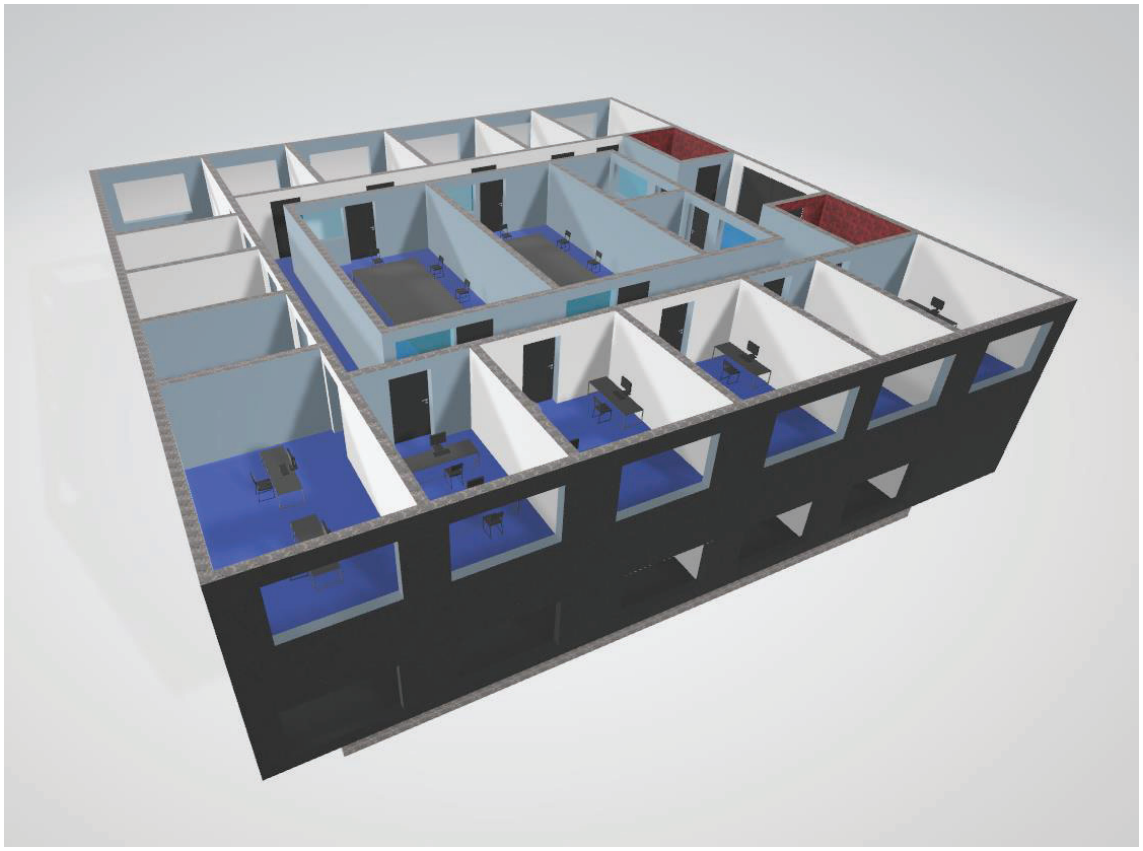

Figura 58 Bloque 2 planta 2. Elaboración propia (2021).

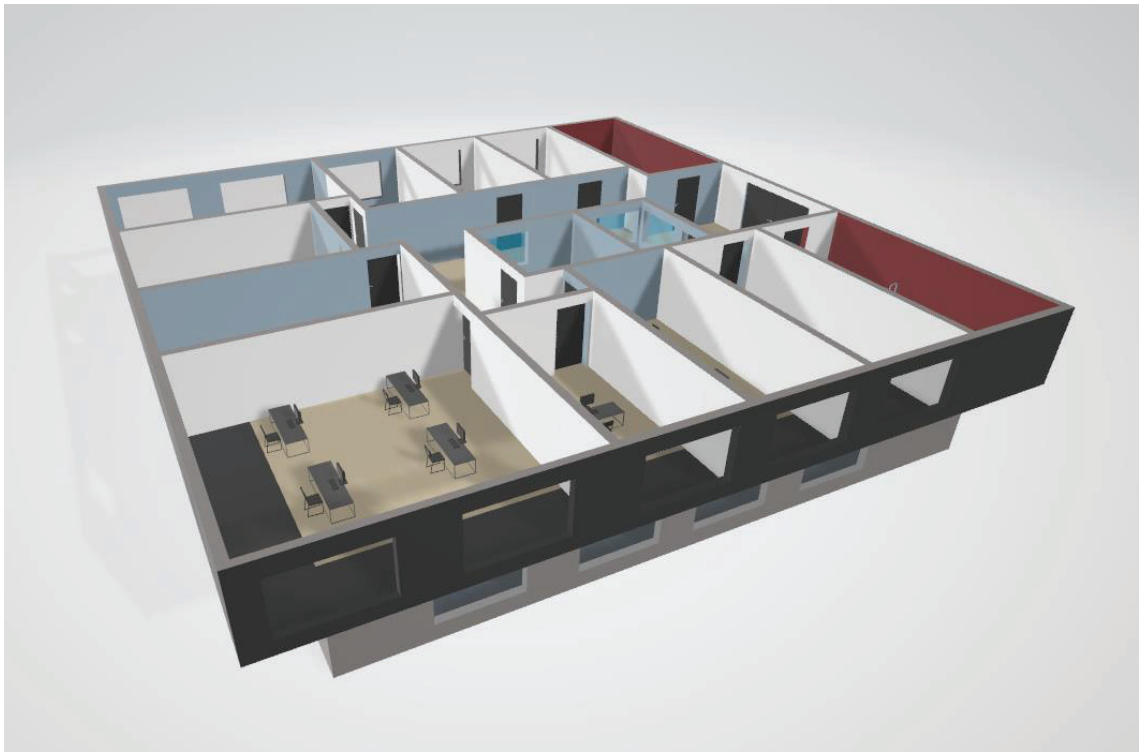

Figura 59 Bloque 2 planta 1. Elaboración propia (2021).

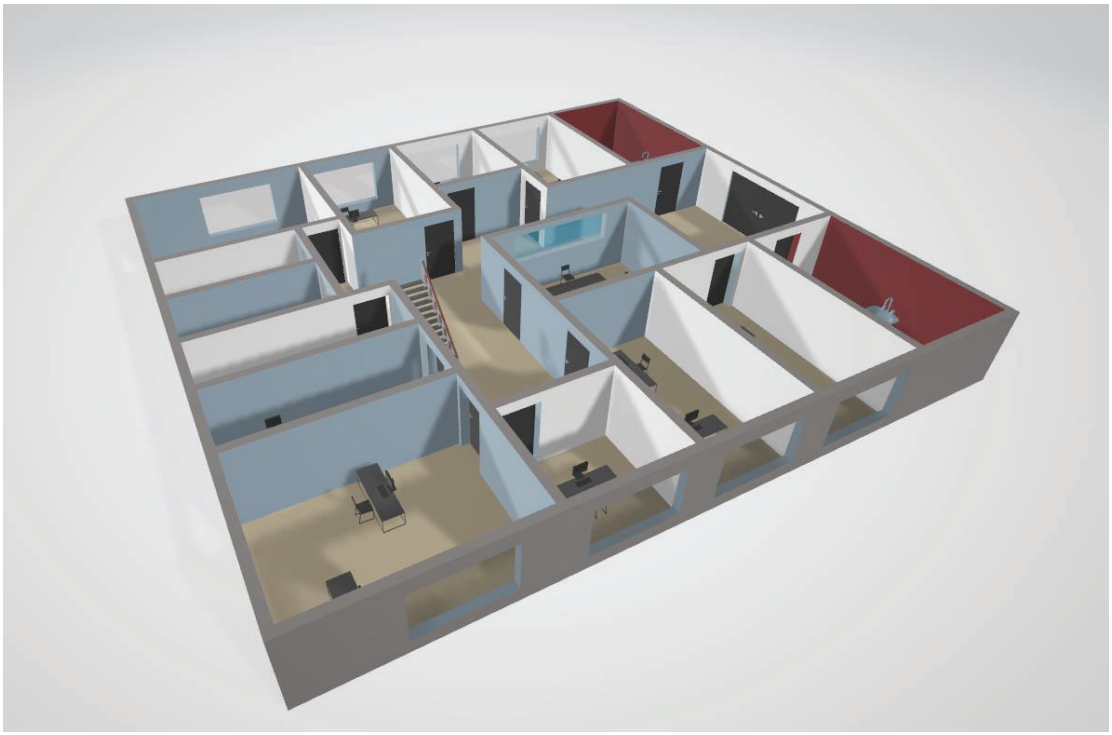

Figura 60 Bloque 2 planta 0. Elaboración propia (2021).

2.9.2 Modelo 3D bloque 3

Esta sección contendrá las imágenes que representan el modelo de dicho bloque. Se puede observar en cada figura cada una de las plantas que será representada en la aplicación.

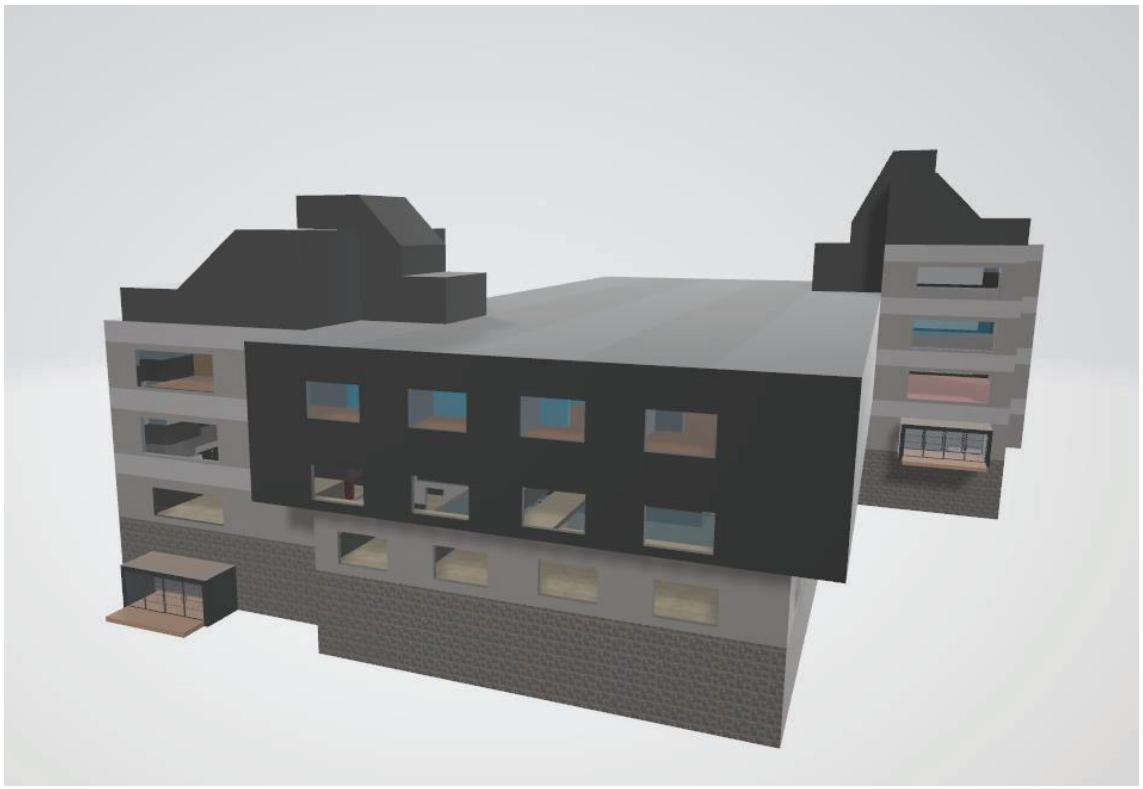

Figura 61 Bloque 3 completo. Elaboración propia (2021).

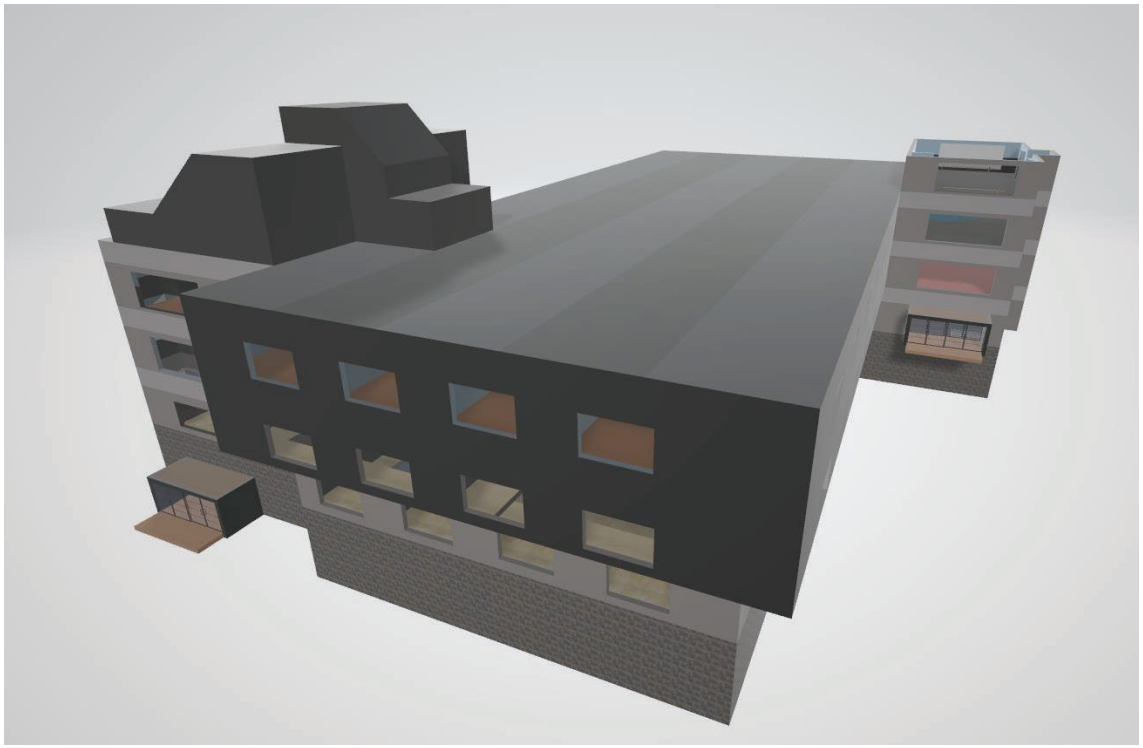

Figura 62 Bloque 3, parte bloque 3-4. Elaboración propia (2021).

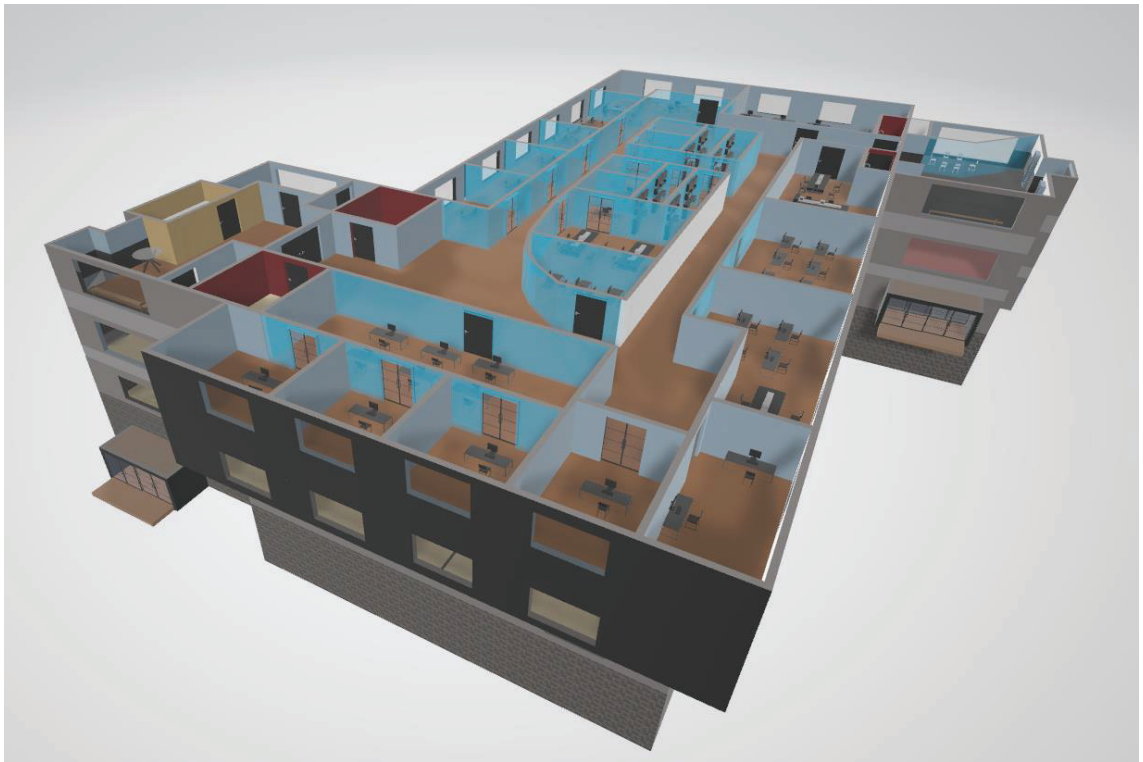

Figura 63 Bloque 3 planta 3. Elaboración propia (2021).

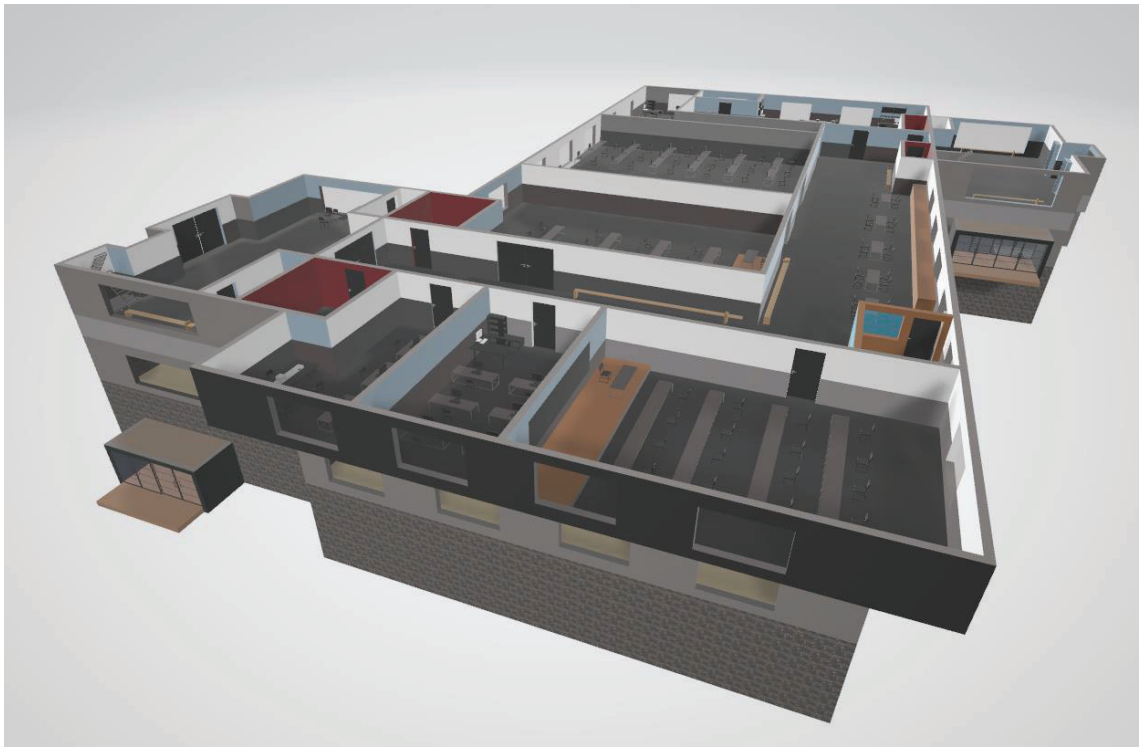

Figura 64Bloque 3 planta 2. Elaboración propia (2021).

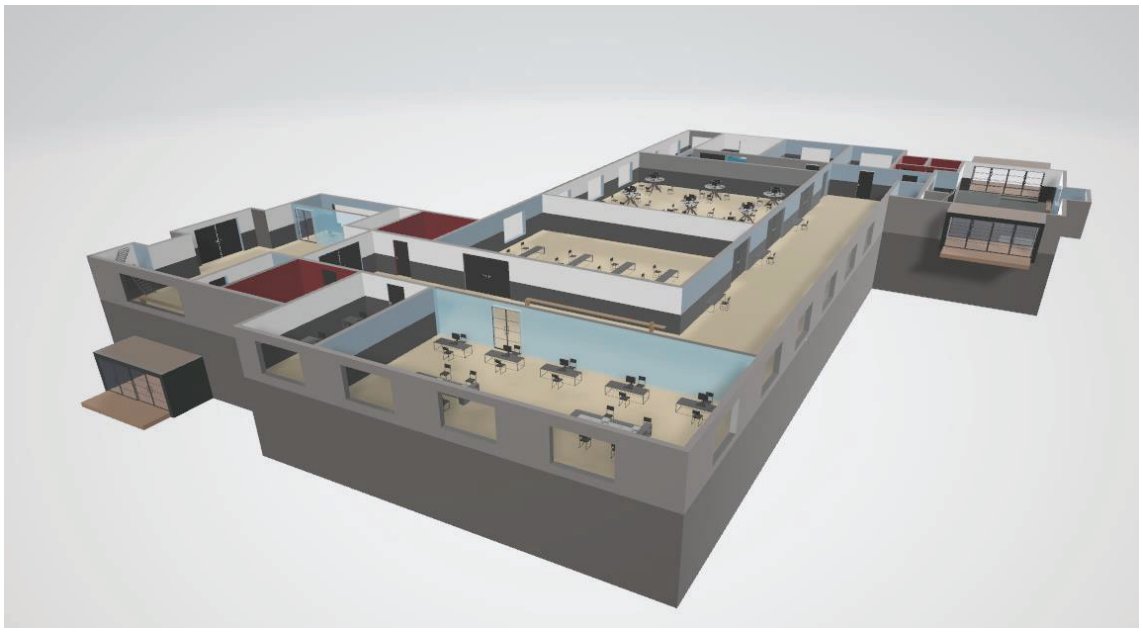

Figura 65Bloque 3 planta 1. Elaboración propia (2021).

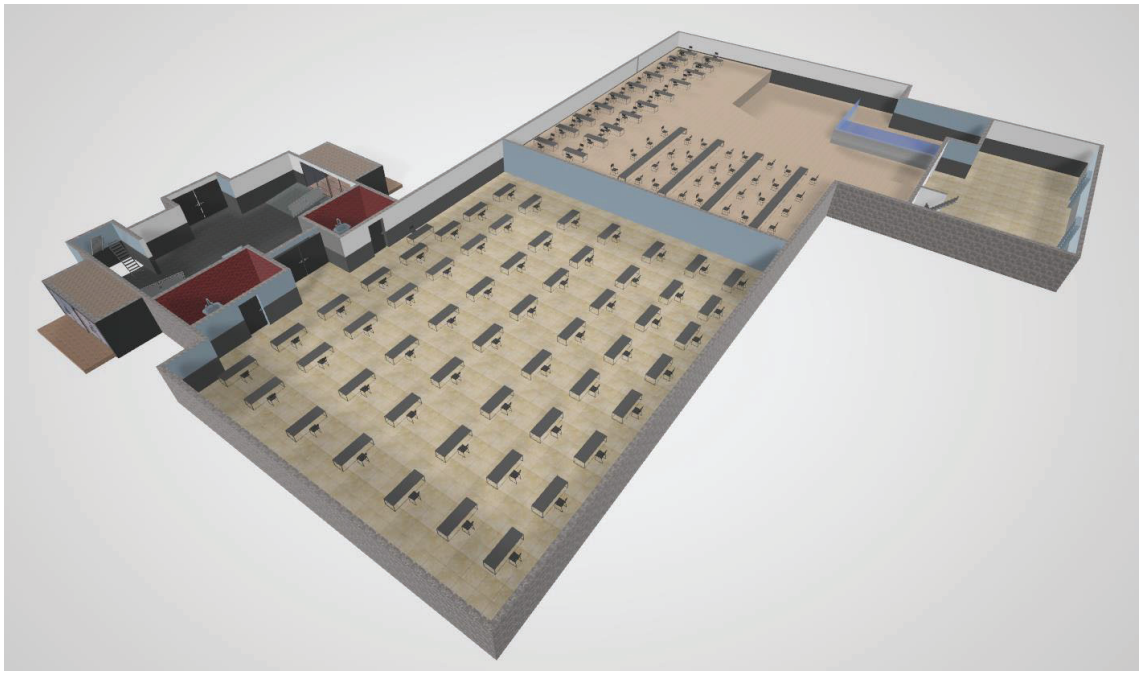

Figura 66 Bloque 3 planta 0. Elaboración propia (2021).

3 Resultados y conclusiones

Los resultados obtenidos han sido positivos, debido a que se ha logrado conseguir un desarrollo y un correcto funcionamiento de todas las funcionalidades que se querían implementar. Además, hacer alusión al aprendizaje de todo este campo, el potencial que tiene y toda la información y estudio necesario que lleva consigo. Por lo que lo más importante, ha sido que los objetivos se han alcanzado y el conocimiento adquirido.

Como conclusión, pese a haber logrado que la aplicación funcione de manera correcta; el diseño de los modelos 3D y conseguir que los *scripts* funcionasen como deberían, es lo que más tiempo ha llevado, de hecho, bastante más de lo esperado, por lo que la finalización del proyecto se retrasó varios meses. Una vez retrasada la entrega del proyecto, se pudo contar con estos meses de margen, se modeló un primer bloque, y como se disponía de tiempo se modeló un segundo bloque. En relación con el tiempo invertido que se necesita, para crear un modelo 3D, depende del bloque. El primero que se modeló fue el bloque 3 (en el cual, cada capa es muy diferente a las demás, por lo que no se pudo reutilizar el diseño de cada una), y se tardó alrededor de cuatro o cinco días por capa (dedicándole cada día más o menos 5 horas), ya que también influía el tiempo de aprendizaje con el *software*. Como se ha mencionado anteriormente, tras retrasar el proyecto, se disponía de tiempo para la realización de otro bloque, por lo que se optó por el bloque 2, ya que tiene muchos lugares de interés, en este caso departamentos. Este último bloque se realizó en un tiempo de más o menos ocho días en total, debido a que el *software* se conocía bastante bien (ya que se invirtió bastante tiempo en aprender el funcionamiento para el primer bloque) y, además, en este caso, si se pudieron reutilizar capas, ya que la mayoría eran iguales, lo que permitió que solo se hicieran cambios mínimos y no hacerlas enteras o desde el principio. Por tanto, el esfuerzo que requiere ampliar el proyecto, con más bloques, depende del tiempo que se tarde en aprender a usar el *software* y del bloque elegido, pero de media, crear un modelo entero de un bloque por primera vez, debería rondar los quince días.

Por otro lado, hay que comentar que uno de los problemas que ha surgido, ha sido que los modelos 3D, en sus versiones iniciales, tenían demasiados polígonos y esto ralentizaba demasiado el movimiento de la cámara, por lo que se ha tenido que hacer una optimización de cada uno.

Dejando a un lado el esfuerzo que supone modelar, hay que mencionar el esfuerzo que supone incluir un modelo 3D en el proyecto, ya que esto se debe repetir para cada capa de un bloque, para ello hay que realizar lo siguiente: invertir tiempo en exportarlo, convertir el objeto *STL* obtenido a un objeto *FBX* (ya que *STL* no es un formato compatible con Unity), importarlo en Unity, ordenar el objeto importado con la estructura mostrada en las figuras Figura 32 y Figura 34 y por último añadir las animaciones y paneles con información de los lugares de interés.

Por último, y en relación con la forma de interactuar el usuario con la aplicación, se cambió el diseño previo que se tenía pensado, a medida que se diseñaba la solución, por uno más estético, ordenado y visual.

4 Líneas futuras del proyecto

En relación con las líneas futuras o evolución del proyecto, se podrían plasmar nuevos objetivos como, por ejemplo:

- Diseñar y añadir los bloques restantes: 1, 4, 5 y 6.
- Los paneles que muestran la información de los lugares de interés, de cada capa, pasen a ser interactivos y puedan mostrar más información, por ejemplo: horarios, contacto, tutorías, etc.
- Mejorar el diseño vidual de la aplicación.

5 Análisis de impacto

En relación con el análisis de impacto, de este proyecto, se pasará a exponer los siguientes contextos:

- **Personal:** El desarrollo de este proyecto me ha proporcionado muchos conocimientos que antes no poseía, saber gestionar el procedimiento para llevarlo a cabo y conocer nuevas herramientas, tecnologías y recursos que no conocía.

- **Empresarial:** los riesgos o variables que no se pueden controlar, de este proyecto, son: la posibilidad de obtener la suficiente información acerca de los recursos necesarios para poder saber si el proyecto es abordable o no, conseguir estos recursos y la falta de tiempo que puede llevar solucionar problemas inesperados.

- **Social:** este proyecto va enfocado a cualquier persona que no conozca la facultad. De este modo, proporciona una visión previa de acerca de cómo está organizado todo, muestra dónde está cada lugar y permite (sin necesidad de desplazarse) visualizar, por dentro y por fuera, cada bloque de la ETSIINF.

- **Económico:** los recursos de carácter humano han sido mínimos, ya que el desarrollo se ha realizado por una persona y esto ha provocado que el tiempo invertido haya sido muy elevado. Por otro lado, los recursos de carácter material han sido ínfimos, ya que una característica del marco que se ha establecido, para el desarrollo, ha sido priorizar los recursos gratuitos.

- **Medioambiental:** el impacto medioambiental que ha tenido este proyecto durante el desarrollo es mínimo, debido a que únicamente se ha necesitado la potencia que consume un ordenador y un dispositivo móvil. Por otro lado, el impacto que puede tener durante su vida útil puede ser bastante positivo, ya que puede reducir las emisiones contaminantes (en caso de que el desplazamiento se realice en vehículos que afectan al medioambiente) y el consumo de energía que genera el proyecto es ínfimo.

- **Cultural:** el hecho de que este proyecto haga uso de la realidad aumentada significa que se ha conseguido dar la oportunidad de observar, sin desplazamientos y de una manera diferente, un lugar en el que nunca se ha estado y proporcionar la posibilidad de ver el interior e interactuar con la propia información. Además, apoya a estas nuevas tecnologías y técnicas de visualizar la información y que en la vida cotidiana cada vez tienen mayor cabida. Por otro lado, también podría tener la oportunidad, de que, en el ámbito educativo, proyectos como este o similares, que utilicen la realidad aumentada, se utilicen para enseñar el potencial que tiene y mostrar información de una manera más dinámica.

6 Bibliografía

[1] Colaboradores de Wikipedia. (2020, 31 octubre). Realidad aumentada. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Realidad_aumentada

[2] DMEXCO. (2018). *Ejemplo de realidad aumentada* [Figura]. DMEXCO. https://dmexco.com/stories/augmented-reality-in-marketing-8-current-examples-2/

[3] Rotondas, T. (2008, 24 agosto). Ejemplo de código QR que dice: "Ojalá vivas tiempos interesantes" [Figura]. Wikipedia. https://es.wikipedia.org/wiki/Archivo:Codigo_QR.svg

[4] PNG Play. (2019, 17 diciembre). Ejemplo de objeto fácil de reconocer mediante escaneo y aplicar características de realidad aumentada. [Figura]. PNG Play. http://www.pngplay.com/image/36925

[5] *Niveles de la Realidad Aumentada - Realidad Aumentada*. (s. f.). Sites Google. https://sites.google.com/site/aumentadatema/niveles-de-la-realidad-aumentada

[6] Lang, A. (2019, 4 septiembre). *Ejemplo de escaneo de código QR* [Figura]. Cnet. https://www.cnet.com/how-to/android-10-share-a-wi-fi-password-in-a-snap-with-a-qr-code/

[7] Salah-ddine, K., & El Filali, Y. (2019, 1 abril). *Ejemplo de interacción con marcador.* [Figura]. ResearchGate. https://www.researchgate.net/figure/example-of-marker-based-AR_fig1_332543647

[8] PIX4D. (2019, 3 julio). *Ejemplo escaneo de objeto y generación de contenido virtual.* [Figura]. PIX4D. https://www.pix4d.com/blog/vr-experience-photogrammetry

[9] Leveille, D (danlev). (2014, 5 junio). *Google Glass.* [Figura]. Wikipedia. https://es.wikipedia.org/wiki/Google_Glass

[10] Asensio, I. (2019, 8 noviembre). *Qué es Unity y para qué sirve*. MasterD. https://www.masterd.es/blog/que-es-unity-3d-tutorial/

[11] Colaboradores de Wikipedia. (2020, 12 noviembre). Unity (motor de videojuego). Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)

[12] Colaboradores de Wikipedia. (2020a, noviembre 12). Blender. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Blender

[13] Google. (s. f.-b). *Choose your development environment | ARCore |*. Google Developers. https://developers.google.com/ar/develop

[14] Wikipedia contributors. (2020, 15 octubre). ARCore. Wikipedia. https://en.wikipedia.org/wiki/ARCore

[15] Apple. (s. f.). *ARKit - Augmented Reality*. Apple Developer. https://developer.apple.com/augmented-reality/arkit/

[16] Viur. (2020, 29 enero). Realidad Aumentada y virtual. https://viur.net/

[17] Google. (2020, 4 noviembre). *Unity overview | ARCore* |. Google Developers. https://developers.google.com/ar/develop/unity

[18] Google. (s. f.-a). *ARCore supported devices* |. Google Developers. https://developers.google.com/ar/discover/supported-devices

[19] Google. (2020b, noviembre 6). ARCore overview |. Google Developers. https://developers.google.com/ar/discover

[20] Linares, I. (2019, 2 septiembre). *Cómo instalar la realidad aumentada de ARCore en móviles Android no compatibles*. El Androide Libre. https://elandroidelibre.elespanol.com/2019/09/como-instalar-realidad-aumentada-arcore-moviles-no-compatibles.html

[21] Gershgorn, D. (2017, 29 agosto). *Google finally admits all you need for augmented reality is a camera*. Quartz. https://qz.com/1064898/google-finally-admits-all-you-need-for-augmented-reality-is-a-camera/

[22] Technologies, U. (s. f.). *'s AR Foundation Framework*. Unity. https://unity.com/unity/features/arfoundation

[23] Unity Technologies. (2021b, abril 10). About ARCore XR Plugin | ARCore XR Plugin | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arcore@4.1/manual/index.html

[24] Unity Technologies. (2021b, abril 10). About ARCore XR Plugin | ARCore XR Plugin | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arcore@4.1/manual/index.html

[25] Unity Technologies. (2021c, abril 10). XR session subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/session-subsystem.html

[26] Unity Technologies. (2021c, abril 10). XR camera subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/camera-subsystem.html

[27] Unity Technologies. (2021d, abril 10). XR depth subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/depth-subsystem.html

[28] Unity Technologies. (2021c, abril 10). Unity - Scripting API: XRInputSubsystem. Unity. https://docs.unity3d.com/2019.4/Documentation/ScriptReference/XR.XRInputSubsystem.html

[29] Unity Technologies. (2021f, abril 10). XR plane subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/plane-subsystem.html

[30] Unity Technologies. (2021g, abril 10). XR raycast subsystem | AR Subsystems | 4.1.7. Unity.

https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/raycasting-subsystem.html

[31] Unity Technologies. (2021d, abril 10). XR anchor subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/anchor-subsystem.html

[32] Unity Technologies. (2021g, abril 10). XR face subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/face-tracking.html

[33] Unity Technologies. (2021h, abril 10). XR image tracking subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/image-tracking.html

[34] Unity Technologies. (2021g, abril 10). XR environment probe subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/environment-probe-subsystem.html

[35] Unity Technologies. (2021c, abril 10). Occlusion subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/occlusion-subsystem.html

[36] Unity Technologies. (2021k, abril 10). XR object tracking subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/object-tracking.html

[37] Unity Technologies. (2021l, abril 10). XR participant subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/participant-subsystem.html

[38] Unity Technologies. (2021k, abril 10). XR mesh subsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/manual/mesh-subsystem.html

[39] Unity Technologies. (2021c, abril 10). Class XRHumanBodySubsystem | AR Subsystems | 4.1.7. Unity. https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@4.1/api/UnityEngine.XR.ARSubsystems.XRHumanBodySubsystem.html

[40] Colaboradores de Wikipedia. (2021, 20 marzo). *SketchUp*. Wikipedia. https://es.wikipedia.org/wiki/SketchUp

[41] GitHub. (2018, 8 mayo). ARCore Android SDK. https://github.com/google-ar/arcore-android-sdk/tree/master/tools/arcoreimg

[42] Google. (2018, 3 mayo). The arcoreimg tool | ARCore |. Google Developers. https://developers.google.com/ar/develop/c/augmented-images/arcoreimg

[43] Google. (2020a, septiembre 1). Augmented Images for AR Foundation | ARCore |. Google Developers. https://developers.google.com/ar/develop/unity-arf/augmented-images

[44] Unity Technologies. (2021r, junio 14). Unity - Scripting API: MonoBehaviour. Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[45] Unity Technologies. (2021s, junio 14). Unity - Scripting API: MonoBehaviour.Awake(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html

[46] Unity Technologies. (2021t, junio 14). Unity - Scripting API: MonoBehaviour.Start(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html

[47] Unity Technologies. (2021u, junio 14). Unity - Scripting API: MonoBehaviour.Update(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

[48] Unity Technologies. (2021t, junio 14). Unity - Scripting API: MonoBehaviour.FixedUpdate(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html

[49] Unity Technologies. (2021u, junio 14). Unity - Scripting API: MonoBehaviour.OnDisable(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnDisable.html

[50] Unity Technologies. (2021v, junio 14). Unity - Scripting API: MonoBehaviour.OnEnable(). Unity. https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnEnable.html

[51] Download Android Studio and SDK tools |. (s. f.). Android Developers. https://developer.android.com/studio

[52] Oracle. (s. f.). Java SE Development Kit 8 Downloads. https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

[53] SDK Platform Tools release notes |. (s. f.). Android Developers. https://developer.android.com/studio/releases/platform-tools

[54] Microsoft. (2016, 14 abril). Download Visual Studio Code - Mac, Linux, Windows. https://code.visualstudio.com/download

7 Anexos

7.1 Preparación previa

En relación con la preparación previa al proceso de compilación y exportación del archivo *apk* y *unitypackage*, se explicará cómo preparar todo previamente.

En primer lugar, para que el proyecto se pueda desarrollar, lo que se debe instalar, importar o utilizar deberá ser:

- Editor Unity 2019.4.1f1.
- *SDK AR Foundation* y *ARCore XR Plugin.*
- *SDK Android* y *JDK* (se explicará en la siguiente sección).
- *SketchUp Free.*

Adicionalmente, y de forma opcional, en este proyecto se ha usado el editor de código *Visual Studio Code*, el cual se puede importar en Unity mediante la siguiente ruta: "Window > Package Manager > All Packages", o descargar mediante el siguiente enlace. [54]

En relación con la estructura de directorios, como se puede observar en la figura Figura 55, cada carpeta contendrá la siguiente información:

- "Animations": animaciones de los modelos 3D (al aparecer y desaparecer), de los cilindros (su función es decorativa) y de los paneles de información.
- "AR": marcadores, objeto *XR Reference Image Library,* llamado *Multiple,* y objetos *Prefab AR Session Origin* y *AR Session*.
- "Canvas": objeto *Prefab Canvas.*
- "Materials": texturas de los modelos 3D,
- "Objects": modelos 3D.
- "Plugins": directorio autogenerado que contiene los archivos necesarios para *Android*.
- "Resources": elementos de los *canvas* de la pantalla inicial y del tutorial.
- "Scenes": escena del proyecto.
- "Scripts": código fuente del proyecto.
- "Sounds": detalle añadido. Pista de audio para la interacción con un bloque y cambiar de capa.
- "StreamingAssets": carpeta autogenerada.
- "XR": carpeta autogenerada.
- "ARCoreSettings": archivo autogenerado.

7.2 Compilación del proyecto, exportación del archivo *apk y unitypackage*

7.2.1 Compilación

En relación con la compilación del proyecto, este se compila automáticamente cada vez que se produce algún cambio, si hay algún error, Unity no permite ni ejecutar ni exportar el proyecto.

7.2.2 Exportación archivo *apk*

Por otro lado, acerca de la exportación del archivo *apk* hacia *Android,* primero será necesario acceder a la siguiente ruta en Unity: "Edit > Preferences > External Tools > Android" e importar el *SDK Android* y *JDK,* para poder exportar la aplicación hacia los dispositivos móviles que posean este sistema operativo. Para ello, será necesario descargar *Android Studio* en el siguiente enlace [51], para poder extraer la ruta donde se encuentra el *SDK*. En este caso, un ejemplo de ruta es el siguiente: "C:\Users\Christian\AppData\Local\Android\sdk". Además, si Unity no instala el *JDK* de forma automática, *se* deberá importar la versión *JDK* 1.8, debido a que Unity está basado en esta y se puede encontrar en el siguiente enlace [52]. Una vez importado lo necesario, se debe cambiar la plataforma a *Android*, para ello se seguirá la siguiente ruta en Unity: "File > Build Settings > Android > Switch Platform".

Una vez realizado el proceso anterior, se deberá permitir la instalación de aplicaciones desconocidas en el dispositivo móvil, para ello habrá que acceder a la siguiente ruta: "Ajustes > Aplicaciones > Acceso especial > Instalar apps desconocidas > Mis archivos > Permitir desde esta fuente". Cuando se haya habilitado esta opción, se deberán activar las opciones de desarrollador para, más adelante, poder habilitar la depuración por *USB*, para ello habrá que acceder a la siguiente ruta: "Ajustes > Acerca del teléfono > Información de software" y pulsar varias veces sobre el número de compilación hasta que se activen estas opciones. Una vez activadas se seguirá la siguiente ruta para activar la depuración: "Ajustes > Opciones de desarrollador > Depuración por USB". En este punto el dispositivo móvil ya está listo para poder instalar un archivo *apk.*

Seguidamente se conectará el dispositivo móvil al ordenador mediante un cable *USB*. Para verificar que está autorizado, se usará la herramienta *adb*, la cual se puede encontrar en el siguiente enlace [53]. Se escribirá la siguiente sentencia: "adb devices" y se deberá obtener un obtener un *identificador* y la palabra *devices*, esto indicará que está autorizado, en caso contrario, puede que no aparezca ningún id asociado a este dispositivo (debido a que esté mal conectado, problemas del cable, depuración de USB no activada, etc.) o aparezca la palabra *unauthorized* al lado del identificador (aparecerá una notificación en el dispositivo y habrá que pulsar "permitir" para autorizarlo y solucionar el problema).

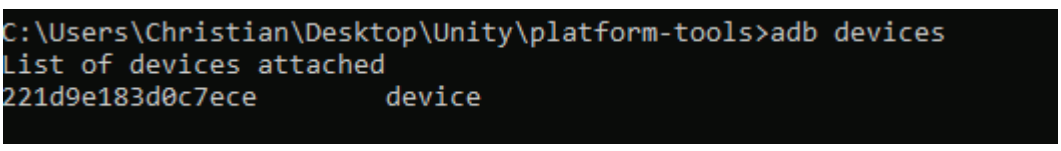

Figura 67 Herramienta adb, resultado correcto. Elaboración propia (2021).

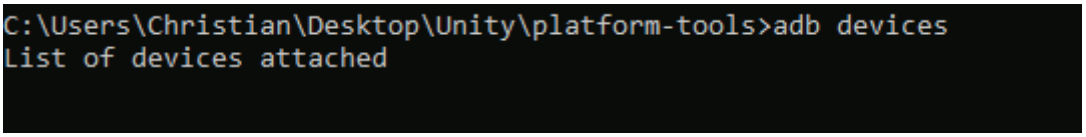

Figura 68 Herramienta adb, primera opción resultado incorrecto. Elaboración propia (2021).

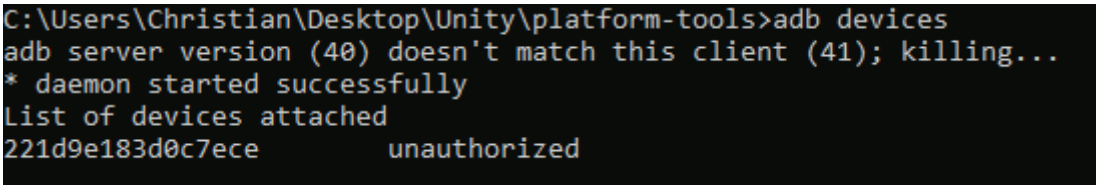

Figura 69 Herramienta adb, segunda opción resultado incorrecto. Elaboración propia (2021).

En este punto, se podrá exportar sin problemas el archivo *apk*. Para ello, desde Unity, habrá que acceder a la siguiente ruta: "File > Build Settings > Build and Run" o "File > Build and Run", se empezará a exportar la aplicación y se iniciará de forma automática en el dispositivo móvil.

Como información adicional, si se inicia la aplicación, y se mantiene conectado el dispositivo móvil al ordenador, mediante la herramienta *adb* y la sentencia "adb logcat | findstr -i unity" se puede obtener una traza del proceso de ejecución y así poder depurar si se produce algún fallo. Véase la siguiente imagen:

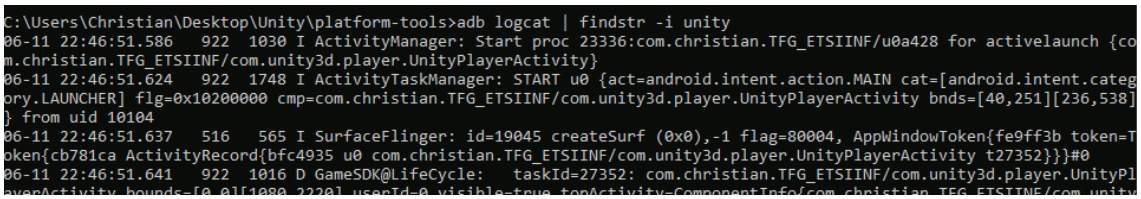

Figura 70 Herramienta adb, ejemplo depuración. Elaboración propia (2021).

Por otro lado, en la siguiente imagen se puede observar la aplicación instalada:

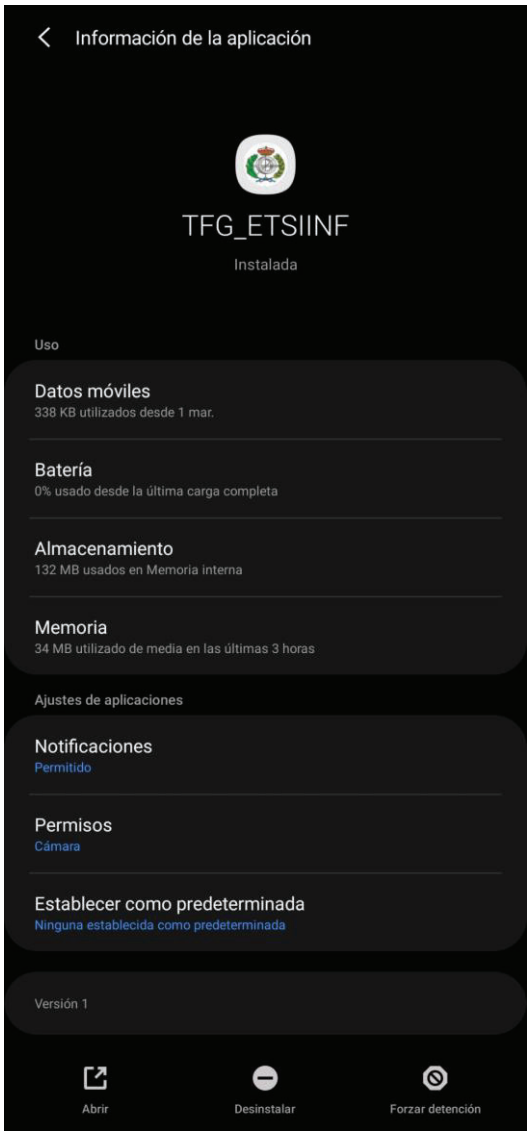

Figura 71 Aplicación instalada. Elaboración propia (2021).

7.2.3 Exportación archivo *unitypackage*

Por último, en relación con la exportación de proyecto y creación del archivo unitypackage, se deberá seguir la siguiente ruta: "Assets > Export package", marcar las casillas *Include dependences* y *All*, y pulsar *Export*.

7.3 Manual de usuario

El usuario cuando inicie la aplicación visualizará la siguiente pantalla, la cual poseerá dos botones y la siguiente información:

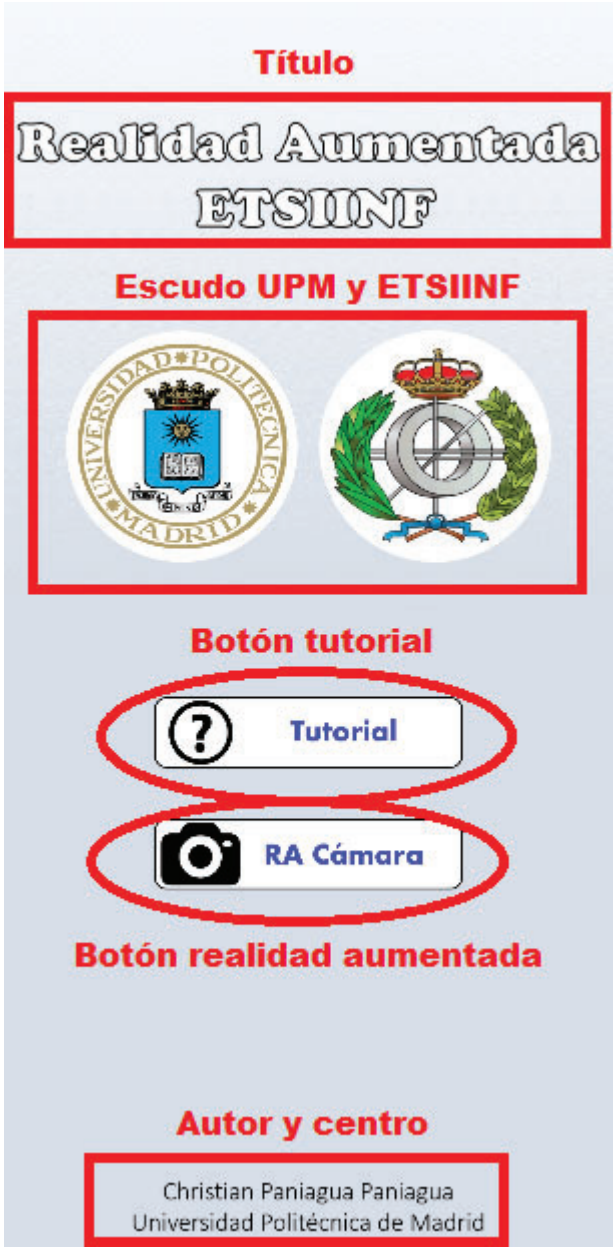

Figura 72 Pantalla inicial. Elaboración propia (2021).

Si se pulsa el botón que activa el tutorial aparecerá la primera pantalla de este:

Figura 73 Primera pantalla tutorial. Elaboración propia (2021).

Si se pulsa el botón para ir hacia atrás volverá a la pantalla inicial. Por otro lado, si se pulsa el botón para ir hacia la siguiente pantalla, se mostrará la segunda parte del tutorial.

La siguiente imagen muestra la segunda pantalla del tutorial:

Figura 74 Segunda pantalla tutorial. Elaboración propia (2021).

Si se pulsa el botón para ir hacia atrás volverá a la primera pantalla del tutorial. Por otro lado, si se pulsa el botón para ir hacia la siguiente pantalla, se mostrará la tercera parte del tutorial.

La siguiente imagen muestra la tercera pantalla del tutorial:

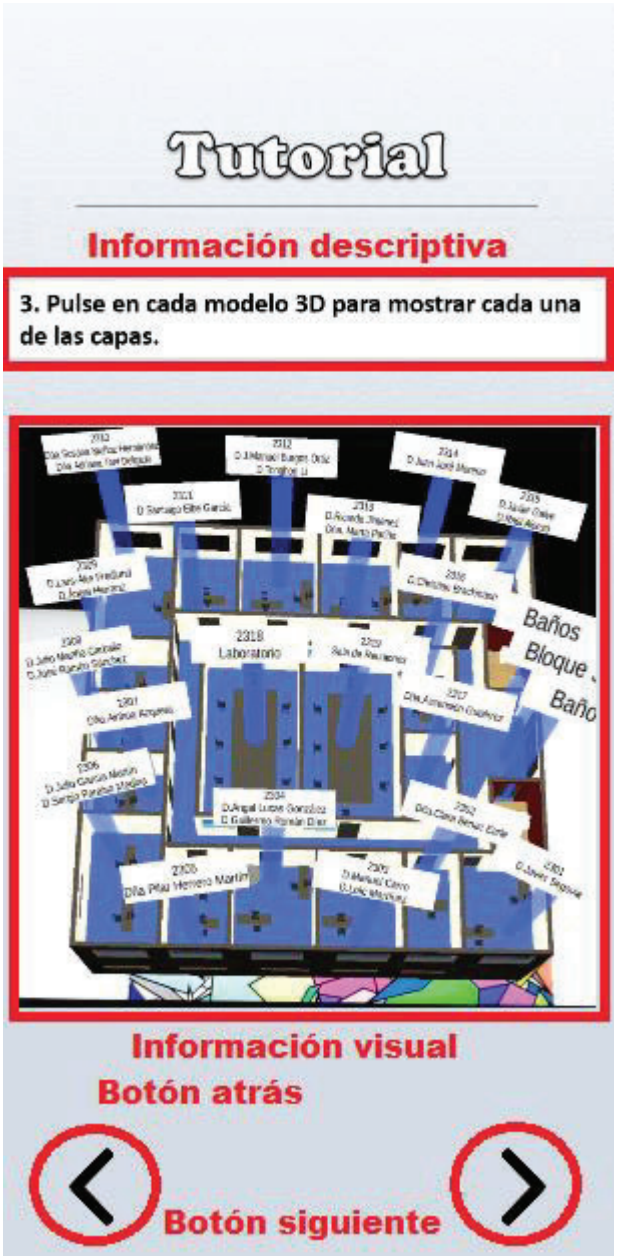

Figura 75 Tercera pantalla tutorial. Elaboración propia (2021).

Si se pulsa el botón para ir hacia atrás volverá a la segunda pantalla del tutorial. Por otro lado, si se pulsa el botón para ir hacia la siguiente pantalla, se mostrará la cuarta parte del tutorial.

La siguiente imagen muestra la cuarta pantalla del tutorial:

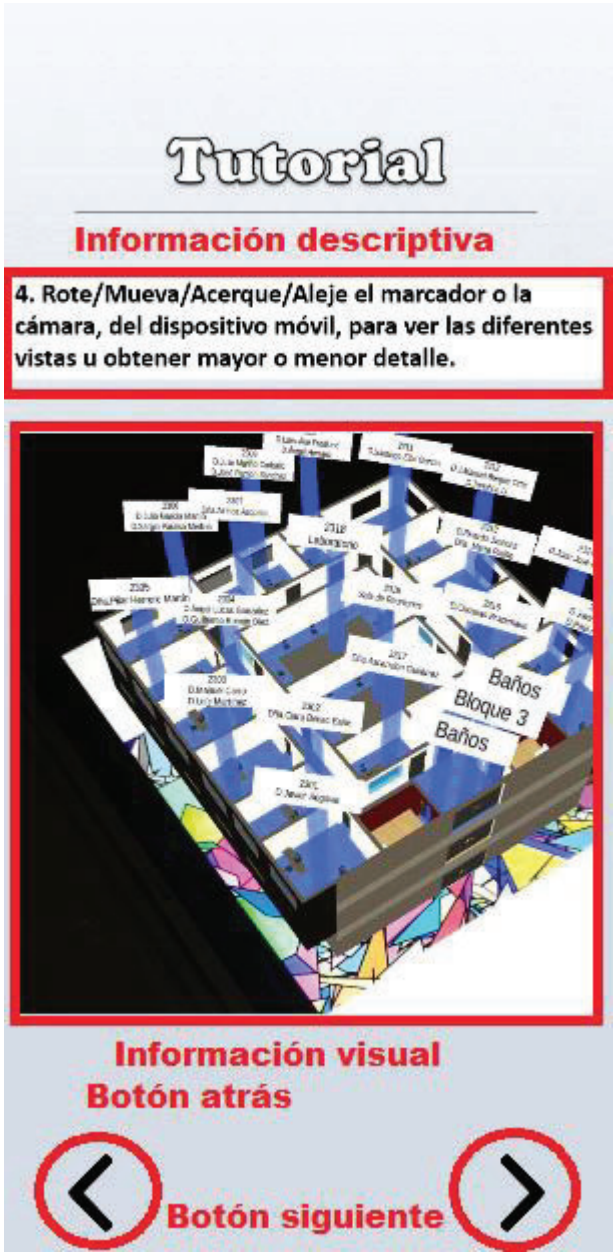

Figura 76 Cuarta pantalla tutorial. Elaboración propia (2021).

Si se pulsa el botón para ir hacia atrás volverá a la tercera pantalla del tutorial. Por otro lado, si se pulsa el botón para ir hacia la siguiente pantalla, se mostrará la última parte del tutorial.

La siguiente imagen muestra la última pantalla del tutorial:

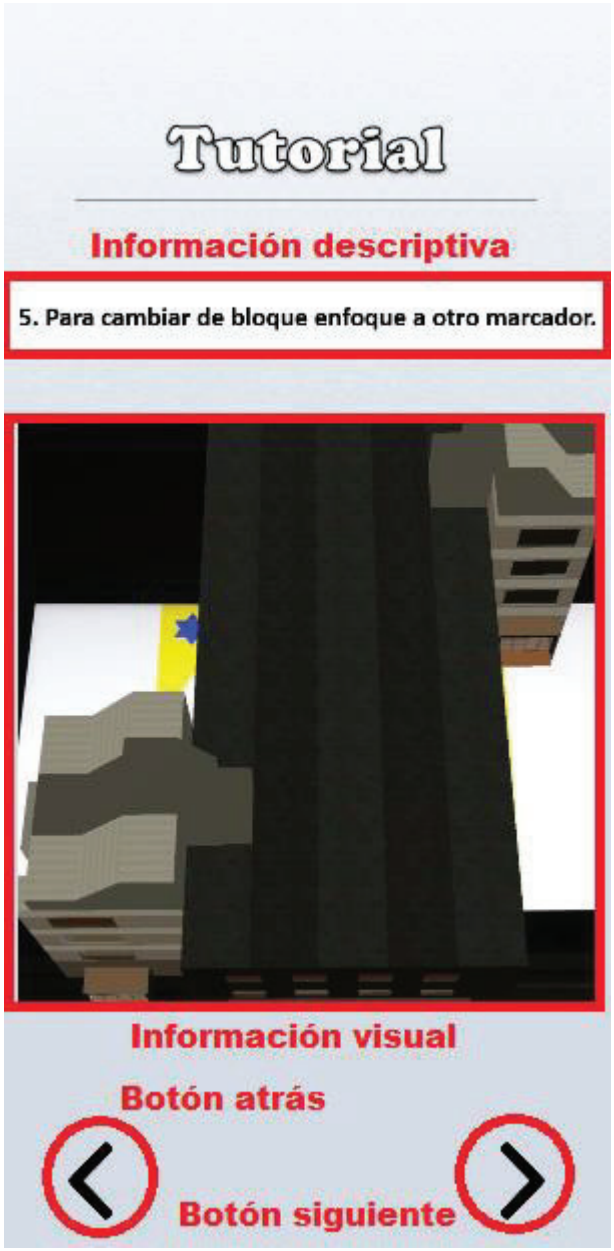

Figura 77 Última pantalla tutorial. Elaboración propia (2021).

Si se pulsa el botón para ir hacia atrás volverá a la cuarta pantalla del tutorial. Por otro lado, si se pulsa el botón para ir hacia la siguiente pantalla, se mostrará la pantalla inicial.

En relación con el botón que da acceso a la realidad aumentada, si se pulsa aparecerá lo que capta la cámara. Si se detecta un marcador, se generará el modelo 3D. Además, hay que procurar que, a la hora de detectarlo, las condiciones de luminosidad sean óptimas, evitar reflejos y realizar el escaneo desde una distancia moderada, para así evitar que el tiempo de detección y reconocimiento se incremente.

Si se pulsa sobre un modelo 3D, se cambiará de capa. Véase, en las siguientes figuras, un ejemplo de uso:

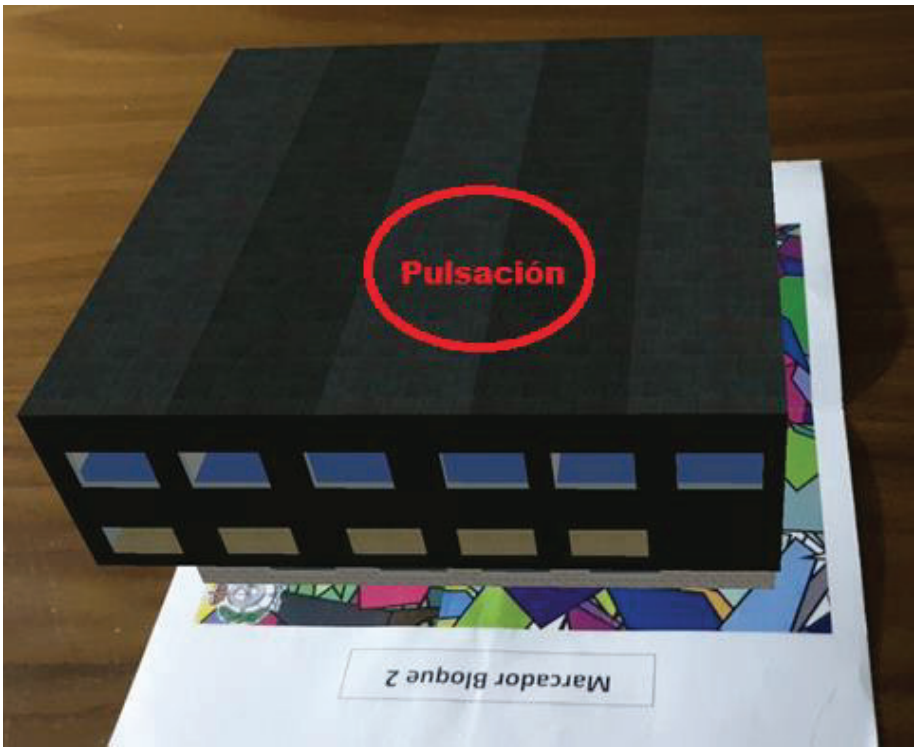

Figura 78 Ejemplo de escaneo y cambio de capa. Elaboración propia (2021).

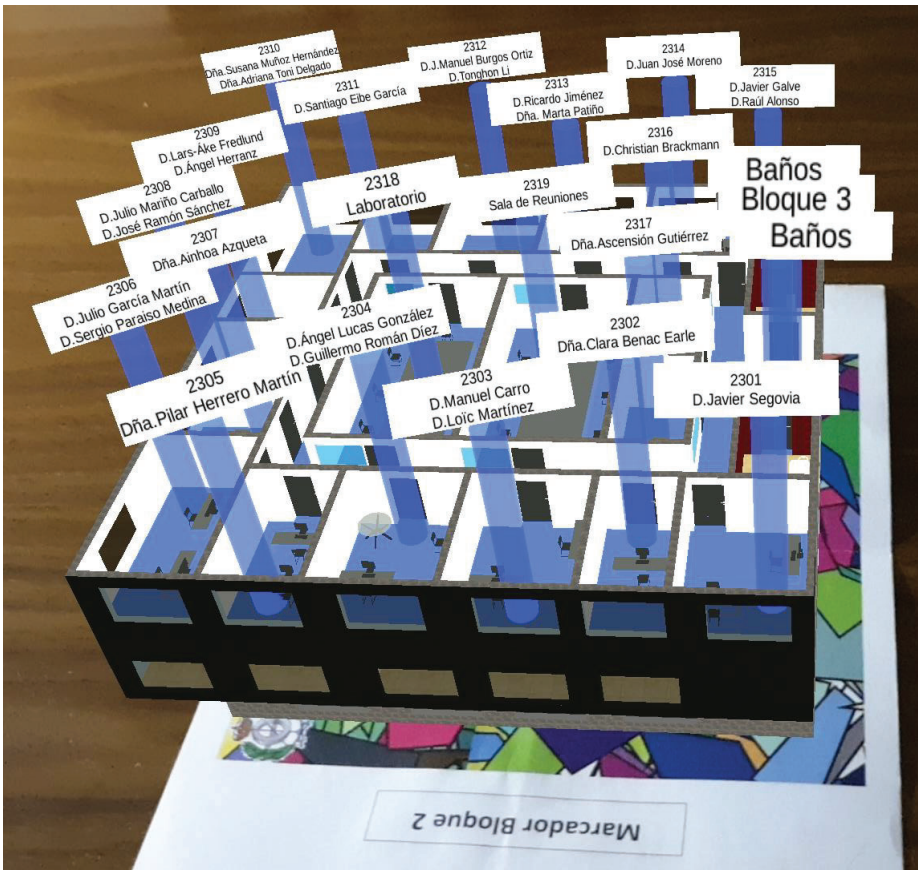

Figura 79 Ejemplo de escaneo y visualización de nueva capa. Elaboración propia (2021).

Si se detecta otro marcador diferente se generará su respetivo modelo 3D. Se muestra en la siguiente figura:

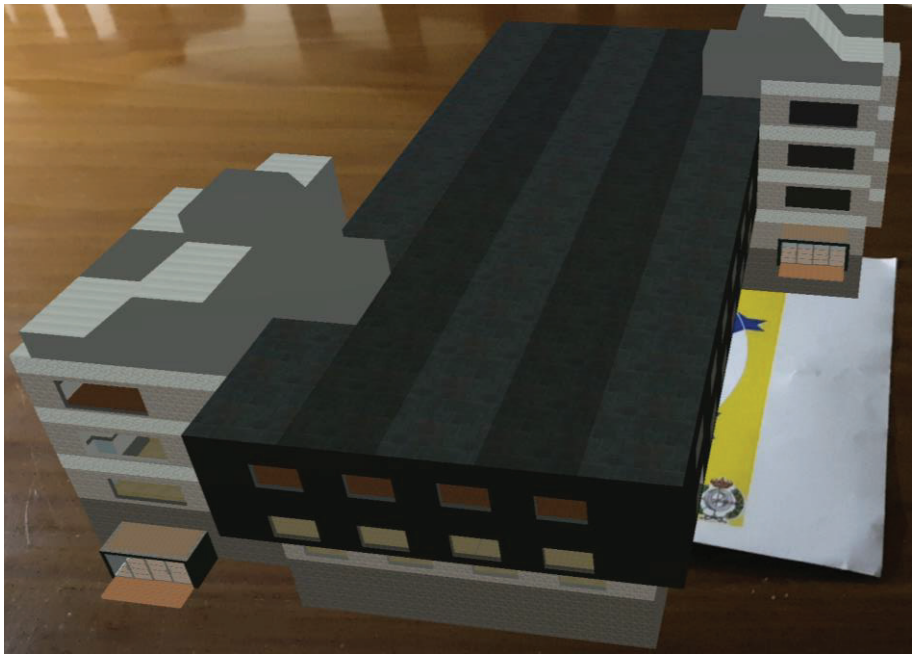

Figura 80 Ejemplo escaneo marcador diferente. Elaboración propia (2021).

Por último, si se detectan varios marcadores a la vez, se visualizarán los modelos. Además, también se podrá interactuar con cada uno de ellos. Véase la siguiente imagen:

Figura 81 Ejemplo escaneo varios marcadores. Elaboración propia (2021).

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Fri Jun 25 14:41:06 CEST 2021
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)