TakeMeHome3D

Trabajo de Fin de Grado

Ingeniería Informática

Julio de 2023

Autor

César López Feo

Tutores

María José Polo Martín

Jesús Lucas Natal

Walter Vinci

Dª María José Polo Martín profesora del departamento de Informática y Automática de la Universidad de Salamanca, D. Jesús Lucas Natal, Software Specialist en HP SCDS y D. Walter Vinci, Software Master Specialist en HP SCDS

CERTIFICAN:

Que el trabajo titulado “TakeMeHome3D” ha sido realizado por César López Feo, con DNI 71043109L, y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 29 de junio de 2023

Dª María José Polo Martín
Dpto. Informática y Automática
Universidad de Salamanca

POLO MARTIN MARIA JOSE - 07847347T
Firmado digitalmente por POLO MARTIN MARIA JOSE - 07847347T
Fecha: 2023.06.29 18:32:38 +02'00'

D. Jesús Lucas Natal
Software Specialist
HP SCDS

Jesús Lucas Natal
Firmado digitalmente por Jesús Lucas Natal
Fecha: 2023.06.29 13:35:04 +02'00'

D. Walter Vinci
Software Master Specialist
HP SCDS

Firmado por Walter Vinci el día 29/06/2023 con un certificado emitido por HP SmartCard CA

Resumen En este proyecto se realiza el diseño e implementación de un sistema para la captura de vídeo o imágenes para el entrenamiento de un modelo de inteligencia artificial utilizando la cámara de un dispositivo móvil. Las imágenes capturadas se procesarán mediante técnicas NeRF(Neural Radiance Fields) para la creación de un modelo 3D que se utilizará para poder visualizar aquello que se haya capturado en el mundo real mediante Realidad Aumentada.
Las técnicas NeRF pueden ser utilizadas para la generación de imágenes de una escena no antes vistas, es decir, dadas varias imágenes de un objeto desde ciertos ángulos, el modelo es capaz de generar nuevas imágenes de ese mismo objeto desde perspectivas no antes vistas. Esto, a su vez, puede ser utilizado para la creación de una representación 3D del objeto o de la escena analizados.
El objetivo es poder, de una forma sencilla, obtener un objeto 3D para su visualización en diferentes entornos. Por ejemplo, capturar una silla para después poder visualizarla en el entorno de nuestro hogar antes de proceder a su compra.

Keywords: Inteligencia Artificial, NeRF (Neural Radiance Fields), Realidad Aumentada, modelo 3D.

Abstract This project designs and implements a system for capturing video or images for training an artificial intelligence model using a cell phone camera. The captured images will be processed using NeRF (Neural Radiance Fields) techniques for the creation of a 3D model that will be used to visualize what has been captured in the real world through Augmented Reality.
NeRF techniques can be used to generate images of a scene not seen before, in other words, given several images of an object from certain angles, the model is able to generate new images of the same object from perspectives not seen before. This, in turn, can be used for the creation of a 3D representation of the analyzed object or scene.
The goal is to be able to, in a simple way, obtain a 3D object for visualization in different environments. For example, capturing a chair and then being able to visualize it in our home environment before proceeding to purchase it.

Keywords: Artificial Intelligence, NeRF (Neural Radiance Fields), Augmented Reality, 3D model.

Tabla de contenidos

1. Introducción y antecedentes

Este proyecto forma parte del marco del Observatorio HP 2022-2023 de la empresa HP SCDS en colaboración con la Universidad de Salamanca.

1.1 Fotogrametría

La fotogrametría es una técnica que permite reconstruir objetos 3D a partir de una serie de imágenes.

A partir de una imagen se puede obtener información bidimensional de un objeto [1]. Si añadimos una segunda imagen que tenga algún punto en común con la primera podemos llegar a extraer información tridimensional. La técnica se aprovecha de estos puntos comunes para calcular las distancias e ir creando una nube de puntos en base al contenido de las imágenes.

La nube de puntos obtenida representa el modelo 3D y puede ser transformada en una malla de triángulos para ser exportada. En la figura 1 se puede ver un ejemplo de una reconstrucción de un objeto 3D, situado en el centro de la imagen, junto con las imágenes utilizadas, situadas alrededor del objeto en sus posiciones correspondientes.

Algunos de los usos de la fotogrametría son la virtualización de esculturas, edificios históricos y objetos antiguos, también puede ser utilizada para crear modelos tridimensionales de objetos para impresión 3D [2]. Uno de los usos más habituales de esta técnica es para la obtención de información geográfica con fines catastrales, urbanísticos, topográficos, etc [3], esto se puede ver en la figura 2.

Figura 1 Generación de objetos 3D con fotogrametría

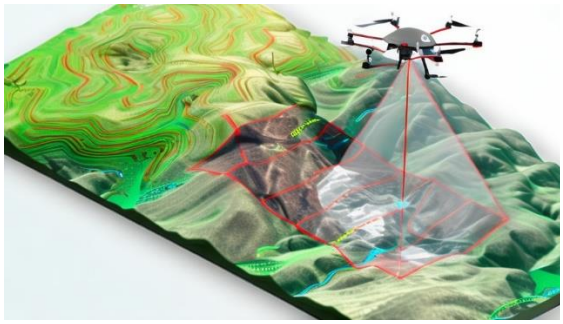


Figura 2 Fotogrametría aplicada a recolección de datos topográficos

Uno de los problemas de esta técnica es que para encontrar puntos comunes en más de una fotografía es necesario que las imágenes tengan condiciones muy similares de iluminación. Además, los objetos transparentes o con superficies reflectantes no pueden ser escaneados ya que producen malos resultados.

1.2 Estudio de aplicaciones similares

Se ha estudiado la aplicación de móvil del IKEA[4] debido a que tiene ciertas similitudes con este proyecto. Esta app cuenta con una funcionalidad llamada *IKEA Kreativ que* permite probar cómo quedarían los diferentes productos de la compañía utilizando el propio dispositivo móvil.

Para ello, la aplicación proporciona varios escenarios virtuales donde se pueden colocar diferentes elementos como muebles, mesas o sofás. En la figura 3 se puede ver uno de estos espacios donde se han colocado varios de estos objetos.

Figura 3 Escenario virtual app IKEA

La parte más relevante de la aplicación es que también permite escanear habitaciones reales para crear espacios virtuales similares al anterior. Para ello, es necesario tomar una serie de imágenes de la sala que se desea escanear desde diferentes perspectivas. Para indicar al usuario como tomar las imágenes de forma correcta se hace uso de realidad aumentada.

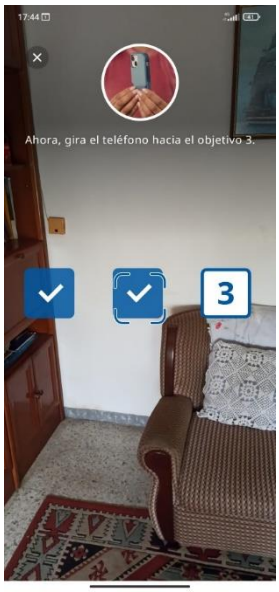


Figura 4 IKEA app, realidad aumentada

En la figura 4 se puede observar la imagen de la cámara del dispositivo con varios cuadrados superpuestos, estos elementos están anclados al entorno de realidad aumentada y no a la pantalla del dispositivo, por lo que su posición se actualiza en función de la imagen del mundo real capturada en tiempo real.

Las imágenes tomadas se procesan para crear un entorno virtual de la parte escaneada de la sala donde, además, se detectan y segmentan los objetos. Esto se puede ver en la figura 5.

Los elementos detectados por la aplicación pueden ser eliminados para conseguir más espacio y poder situar otros objetos. En la figura 6 se puede ver un ejemplo de uso donde se ha sustituido el sofá de la figura anterior por una mesa.

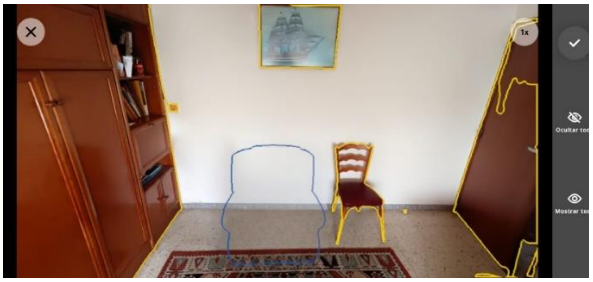


Figura 5 Eliminar objeto en entorno virtual

Figura 6 Añadir objeto en entorno virtual

Los entornos virtuales generados son en realidad una imagen panorámica donde, por medio de inteligencia artificial, se ha calculado el mapa de profundidad de la escena. Esto permite diferenciar elementos como el suelo y las paredes, así como calcular la distancia de los objetos. Información que se utiliza posteriormente para borrar y situar nuevos elementos.

El proyecto que se va a desarrollar tiene ciertas similitudes con esta aplicación, como el uso de realidad aumentada y la capacidad de escanear una escena del mundo real mediante el uso de la cámara.

Por otro lado, la app no permite colocar objetos más allá de los proporcionados por el catálogo del IKEA, lo cual es una gran limitación. Además, los entornos virtuales generados se limitan a una sola perspectiva de la escena.

2. Objetivos del trabajo

El objetivo principal de este proyecto será desarrollar un sistema para el escaneo de objetos y su posterior visualización en realidad aumentada en un dispositivo móvil.

El proyecto se va a dividir en tres partes bien diferenciadas:

- **Una aplicación para dispositivos móviles** que tendrá una función dual: capturar las imágenes y, una vez estén procesadas, poder visualizar los modelos generados en realidad aumentada.
- **Un servicio *REST*** en el lado del servidor para la gestión de los objetos y su almacenamiento en una base de datos.
- **Una *API*** donde se realizará el procesado de las imágenes aplicando las técnicas NeRF. Esta API generará el modelo 3D, idealmente aislando el objeto principal para su mejor visualización mediante realidad aumentada.

2.1 Objetivos principales

Entre los objetivos principales se encuentran:

- **Generación de objetos tridimensionales a partir de un vídeo almacenado**, el usuario podrá iniciar el proceso de generación del modelo 3D de un objeto en base a un vídeo seleccionado desde la galería de su dispositivo.
- **Generar un modelo 3D en base a un conjunto de imágenes de la galería**, el usuario podrá iniciar el proceso de generación del modelo 3D de un objeto en base a un conjunto de imágenes seleccionadas desde la galería de su dispositivo.
- **Gestión de objetos**, el usuario podrá gestionar los objetos. El sistema mostrará una lista con los objetos existentes y el usuario podrá acceder a la información detallada de cualquier objeto. También podrá consultar, eliminar o modificar un objeto existente.
- **Visualización de objetos en realidad aumentada**, el usuario podrá visualizar cualquier objeto 3D previamente generado en realidad aumentada desde su dispositivo móvil. Una vez en la visualización, el usuario podrá rotar, mover o aumentar de tamaño el objeto.

2.2 Objetivos adicionales

Además de los objetivos principales, también se van a definir una serie de objetivos adicionales a fin de ampliar y mejorar el sistema. La implementación de estos objetivos dependerá del avance del proyecto, pudiendo ser introducidos en cualquiera de los diferentes sprints del desarrollo del sistema.

- **Generación de objetos tridimensionales a partir de un vídeo capturado desde la aplicación**, el usuario podrá grabar un objeto directamente desde la aplicación para la generación de un modelo 3D.
- **Generar un modelo 3D en base a imágenes capturadas desde la aplicación**, el usuario podrá tomar una serie de fotos desde la aplicación para la posterior generación de un modelo 3D.
- **Gestión de usuarios,** el usuario podrá registrarse introduciendo una serie de datos. Una vez registrado el usuario podrá iniciar sesión en el sistema. También podrá modificar sus datos o eliminar la cuenta creada. Cada usuario solo tendrá acceso a sus objetos generados.
- **Compartir objetos generados con otros usuarios**, el sistema permitirá a los usuarios compartir sus objetos creados con otros usuarios que utilicen la aplicación.
- **Exportar objeto**, el usuario podrá exportar como archivo el objeto generado para almacenarlo en la propia memoria del dispositivo.
- **Visualizar varios objetos 3D simultáneamente**, el usuario podrá visualizar varios objetos de su lista en realidad aumentada de forma simultánea.
- **Establecer tiempo de entreno del modelo**, a la hora de generar un nuevo modelo 3D, el usuario podrá decidir el tiempo de entreno del algoritmo de inteligencia artificial.
- **Sistema de notificaciones**, el sistema notificará al usuario cuando el modelo 3D del objeto escaneado esté listo para descargarse.

3. Conceptos teóricos

3.1 Realidad Aumentada

La Realidad Aumentada (RA) es una tecnología que permite superponer elementos virtuales a entornos reales. Funciona a través de dispositivos tecnológicos como móviles o gafas que añaden información digital a la realidad que percibe el usuario. Para poder utilizar esta tecnología es necesario disponer de un dispositivo dotado de una cámara, una unidad de procesamiento, una pantalla y tener instalado un determinado software.

La Realidad Aumentada está formada principalmente por dos componentes:

- Por un lado, está el motor gráfico, este es el encargado de renderizar los objetos que se van a superponer sobre la imagen del mundo real. Generando una imagen en dos dimensiones a partir del objeto en 3D [5].

- Por otro lado, se encuentra la visión artificial, esta es la encargada de ubicar los contenidos generados por el motor gráfico en la escena de forma que el resultado sea coherente. Para ello, es necesario analizar el entorno sobre el cual se desea colocar elementos virtuales y tomar puntos referencia, lo que se conoce como seguimiento o *tracking*. Algunos ejemplos serían el seguimiento de imágenes, el seguimiento de superficies o el seguimiento de objetos.

La realidad aumentada requiere del procesamiento en tiempo real de las imágenes y de los objetos para poder garantizar una experiencia fluida. Como resultado, se obtiene una imagen del mundo real con un objeto superpuesto como se puede ver en las figuras 7 y 8.

Figura 7 Realidad aumentada en una tablet

Figura 8 Realidad aumentada en un móvil

3.1.1 Realidad aumentada con ARCore

ARCore[6] es una API diseñada para dispositivos Android que permite la implementación de aplicaciones con realidad aumentada. Esta tecnología está basada en varios conceptos fundamentales:

- Comprensión del entorno, la API puede detectar superficies horizontales o verticales como mesas, suelos o paredes mientras procesa la entrada de la cámara del dispositivo. Estas superficies detectadas se denominan planos. ARCore es capaz de unir objetos virtuales a diferentes planos en una ubicación y orientación fijas.

- Mediante el seguimiento del movimiento, ARCore determina la posición y la orientación del dispositivo en relación con el mundo real, para ello, hace uso de la información visual de la cámara y de la información proporcionada por los sensores del teléfono. También realiza un seguimiento de los objetos virtuales posicionados en la escena para poder mostrarlos con la perspectiva correcta.

- Estimación de la luz, la herramienta entiende la iluminación del entorno. Utilizando esa información es capaz de ajustar los colores de los objetos virtuales haciendo que se vean de forma más realista.

- Interacción del usuario, al tocar la pantalla, ARCore proyecta un rayo desde el punto elegido por el usuario que colisiona con los diferentes elementos con los que se cruza como planos y objetos virtuales permitiendo seleccionarlos e interactuar con ellos[7].

3.2 Volume ray marching (marcha de rayos volumétricos)

Antes de explicar cómo funcionan las técnicas *NeRF* es importante hablar de una técnica de renderizado de gráficos conocida como *volume ray marching*[8]. El renderizado de gráficos es el proceso por el cual a partir de una escena tridimensional se obtienen imágenes bidimensionales desde una cierta perspectiva.

El *volume ray marching* es una técnica de renderizado que consiste en el trazado de rayos desde un punto en el espacio. Estos rayos se hacen pasar por el interior de los objetos tridimensionales de la escena con los que se crucen con el fin de obtener la máxima información volumétrica de los mismos. A partir de esa información se determina el color de los diferentes píxeles de la imagen final.

En la figura 9 se puede ver un rayo atravesando una representación de un objeto tridimensional(1), el rayo se traza desde el píxel que se quiere calcular con este método.

Los puntos de la imagen representan los *voxeles* que conforman el objeto(2). Los *voxeles* son las unidades cúbicas que componen los objetos tridimensionales, son el equivalente a los píxeles, pero en tres dimensiones.

Como se observa en el tercer(3) paso de la imagen, los *voxeles* que atraviesa el rayo son los que se tienen en cuenta para definir el color del píxel final por lo que se tiene que calcular el color de cada uno de ellos, esto se realiza mediante una **función de transferencia F_Θ** que tiene en cuenta el punto de partida del rayo (x, y, z) y el ángulo (θ, ϕ) de este.

Por último, se utilizan los colores calculados previamente para el cálculo del color y opacidad del pixel final(4).

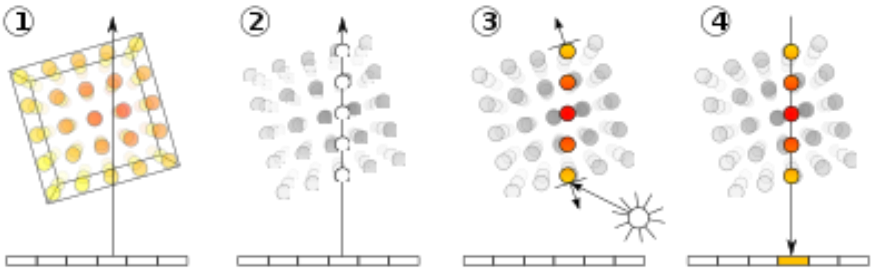


Figura 9 Volume ray marching [8]

Esta técnica se usa actualmente en campos como los videojuegos para el renderizado de nubes volumétricas y partículas para conseguir un mayor realismo. El problema de esta técnica es que es más costosa computacionalmente que otros métodos de renderizado.

3.3 Redes neuronales

Las redes neuronales están formadas por unidades más pequeñas denominadas neuronas. Una **neurona** es una unidad de cálculo capaz de generar un valor de salida en función de una serie de valores de entrada y de los parámetros internos o pesos de la misma. Para ello, realiza una suma ponderada de los datos de entrada y los hace pasar por una función de activación.

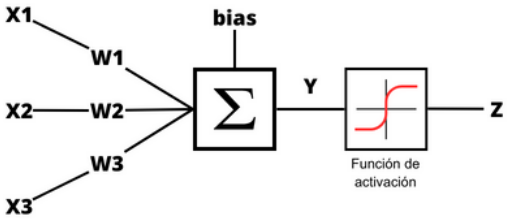


Figura 10 Neurona

En la figura 10 se puede ver un ejemplo de neurona con tres datos de entrada (x1, x2, x3), estos se multiplican por los pesos (w1, w2, w3) y se suman. Adicionalmente, se le añade el *bias*, este es otro de los parámetros internos de la neurona, a veces denominado w_0, que permite sesgar el resultado de la suma ponderada. La operación realizada corresponde con la siguiente fórmula:

$$Y = \sum(x * w) + \text{bias}$$

El resultado anterior se pasa por una función de activación que determinará el valor final de la neurona. La función de activación se utiliza para que el valor de salida se encuentre siempre en el mismo rango de valores. Un ejemplo es la función sigmoide, que devuelve valores entre 0 y 1 en función del valor de Y siguiendo la siguiente fórmula:

$$Z = sigmoide(Y) = \frac{1}{1 - e^{-Y}}$$

Vamos a considerar una red neuronal de una única neurona. Esta necesitará pasar por un proceso de entrenamiento donde se busca calcular los valores de los pesos que resuelvan una determinada tarea.

Para ello, es necesario un ***dataset* de entrenamiento**, es decir, un conjunto de datos en los que ya esté resuelta la tarea que queremos que aprenda la red. Antes de comenzar el entrenamiento, se inicializan los pesos con valores aleatorios.

El proceso de entrenamiento consiste en resolver un problema de optimización y sigue los siguientes pasos de forma iterativa:

1. Se utilizan los datos de entrada del ***dataset* de entrenamiento** para pedirle a la red que haga una predicción, devolviendo así un resultado.

2. El resultado generado se compara con el resultado esperado mediante una **función de coste** que medirá el error cometido por la red. Este valor es el que hay que optimizar, tratando de reducirlo lo máximo posible.

3. Utilizando el método del **descenso del gradiente**. Se calcula la derivada parcial de la función de coste con respecto a cada uno de los pesos para estudiar cómo afectaría al error final modificar cada uno de estos.

4. Se modifica el valor de cada peso en la dirección contraria al resultado del cálculo anterior para reducir el error final.

Con una red con una única neurona solo se pueden resolver problemas muy sencillos, por lo que las redes neuronales que se usan habitualmente suelen estar formadas por múltiples neuronas. Estas se conocen por el nombre de **redes neuronales profundas** y son un conjunto de neuronas organizadas en capas donde, cada neurona de una capa (**n**) tiene como entradas los resultados de todas las neuronas de la capa anterior (**n-1**).

La primera y última de las capas se denominan capas de entrada y de salida respectivamente. Las capas intermedias se denominan capas ocultas, el número de capas ocultas determinará la profundidad de la red neuronal. En la figura 11 se puede ver una esquema de una red neuronal profunda con dos capas ocultas.

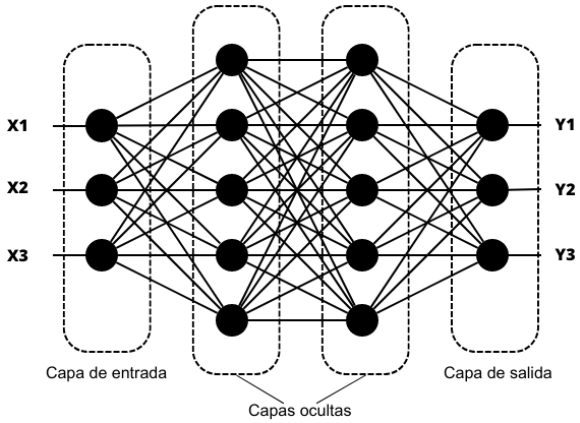


Figura 11 Red neuronal

La información viaja de izquierda a derecha a través de las distintas capas hasta la capa de salida donde se obtendrá un resultado. Una red neuronal se puede considerar como una caja negra, que, dados unos datos de entrada, produce unos datos de salida.

Al igual que con una red de una sola neurona, es necesario pasar por un proceso de entrenamiento donde la red aprenda a resolver una determinada tarea. El proceso se realiza de forma similar, pero en este caso, el cálculo de las derivadas parciales es mucho más complejo. Por ello, se utiliza un algoritmo conocido como ***backpropagation***. Este algoritmo se utiliza para propagar el error hacia atrás a través de las distintas capas mediante el uso de la regla de la cadena de las derivadas, obteniendo un error asociado a cada neurona de la red de forma eficiente. Una vez hecho esto, se puede utilizar el método del **descenso del gradiente** para continuar con el procedimiento explicado anteriormente.

A medida que avance el entrenamiento, la red neuronal irá consiguiendo mejores resultados en la tarea encomendada. Cuanto mayor sea el número de capas ocultas, la red podrá llegar a aprender a resolver problemas más complejos a cambio de aumentar su complejidad y, por tanto, aumentar su tiempo de entrenamiento.

3.4 NeRF: Neural Radiance Fields

Las técnicas *NeRF*[9] realizan un proceso inverso al *volume ray marching,* pero añadiendo el uso de inteligencia artificial y aprendizaje automático.

Como se ha descrito anteriormente, las redes neuronales se pueden entender como una caja negra que, dados unos **datos de entrada**, generan unos **datos de salida**. En este caso, los datos de entrada están formados por las imágenes junto con las posiciones y direcciones de cada una de ellas.

El proceso de entrenamiento de este tipo de modelos sigue una serie de pasos, muchos de ellos similares a los ya explicados en el apartado anterior:

1. Las imágenes de entrada son utilizadas para generar una nube de *voxeles* que conformará el modelo 3D. Este paso es equivalente a aplicar la técnica de *volume ray marching* explicada en el punto 3.2 pero forma inversa y sustituyendo la **función de transferencia (F_Θ)** por una **red neuronal**. Esta red neuronal está formada por múltiples capas densas de neuronas, esto significa que cada neurona de una capa está conectada con todas las neuronas de la capa anterior tal y como se explica en el apartado 3.3.

   Conociendo la posición y dirección de cada imagen, es posible trazar un rayo desde cada píxel como se ve en el apartado (a) de la figura 12. Ahora, la red neuronal va a ser la encargada de determinar los colores y la densidad volumétrica de cada *voxel* que atraviesen estos rayos, esto se puede ver en el apartado (b) de la figura.

   Dicho de otro modo, a partir de una imagen con su posición (x, y, z) y dirección (θ, ϕ), la red neuronal va a determinar el color RGBσ (rojo, verde, azul, densidad de volumen) de cada elemento volumétrico o *voxel* del rayo con dirección (θ, ϕ) y posición inicial (x, y, z).

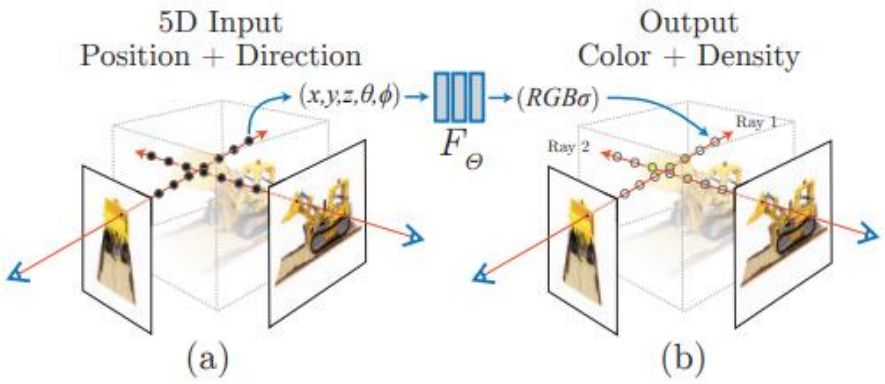


Figura 12 Esquema NeRF

2. Una vez se ha hecho una estimación de los *voxeles* de la escena, se van a utilizar métodos de renderizado volumétrico para generar una serie de imágenes de salida.

3. Las imágenes de salida se van a comparar con las de entrada mediante una **función de coste** calculando así el error de la red.

4. Este error se va a utilizar para la optimización de los pesos de la red neuronal tal y como se describe en el apartado 3.3.

Como resultado, obtendremos un modelo que ha sido capaz de interpretar y aprender la información tridimensional de la escena. Ahora es posible generar imágenes desde nuevas perspectivas, es decir, desde posiciones y direcciones que no estaban en los datos de entrenamiento. Y no solo eso, sino que también es posible extraer la escena tridimensional al completo.

3.4.1 Nerfstudio

La librería Nerfstudio [10] proporciona varios modelos de redes neuronales *NeRF* diferentes que pueden ser usados para renderizar nuevas perspectivas de una escena o reconstruir objetos tridimensionales.

El primer modelo *NeRF* desarrollado es ineficiente y extremadamente lento debido a que no hace uso de las tarjetas gráficas de los diferentes dispositivos. Por lo tanto, en este proyecto se ha optado por utilizar el modelo *Nerfacto* de entre los que proporciona la librería. Este modelo cuenta con múltiples mejoras que consiguen generar mejores resultados mientras que aumentan el rendimiento y disminuyen el tiempo de entrenamiento haciendo uso de las GPUs.

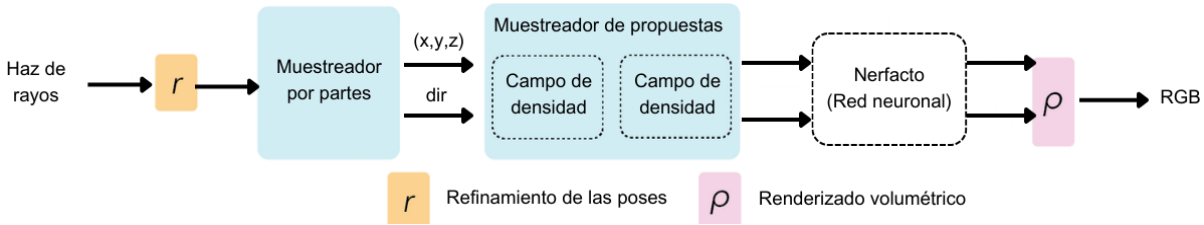


Figura 13 Esquema del modelo Nerfacto

En la figura 13 se puede ver el esquema del modelo *Nerfacto* con todos sus componentes, a continuación, se explica cada uno de ellos:

- **Refinamiento de las poses**
  Uno de los componentes se encarga de refinar las poses de las imágenes de entrada. Debido a que es un problema optimizable se puede usar el propio entrenamiento de la red neuronal para corregir los posibles errores en las posiciones y direcciones calculadas.

- **Muestreador por partes**
  Los muestreadores son los elementos que permiten definir la distancia entre los *voxeles* de cada uno de los rayos que se trazan, es decir, definen el número de muestras a generar por cada rayo y el modo en el que se van a distribuir.

  En este modelo, se utiliza un muestreador por partes para producir el conjunto inicial de puntos de la escena. El muestreador calcula un *voxel* de manera uniforme hasta una distancia de 1 desde la cámara, a partir de ahí, cada *voxel* que se genera está a mayor distancia que el anterior. Esto se puede ver en la figura 14.

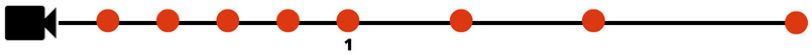


Figura 14 Muestreador por partes

- **Muestreador de propuestas**
  El muestreador de propuestas fija las muestras generadas en el paso anterior que más contribuyen al renderizado final. Estas muestras suelen coincidir con la primera intersección de los rayos con una superficie. Este proceso mejora la calidad de la reconstrucción de la escena sin influir de forma considerable en el rendimiento del modelo.

- **Campo de densidad**
  El campo de densidad se utiliza para guiar en el proceso de muestreo, consiste en tener una representación global de la densidad de toda la escena.

- **Nerfacto (red neuronal)**
  Este componente es el más importante, se trata de una red neuronal multicapa que, como se explicó en el apartado 3.4, a partir de la posición (x, y, z) y dirección (θ, ϕ) de cada rayo, se encargará de predecir cada uno de los *voxeles* que los muestreadores consideren necesarios.

- **Renderizado volumétrico**
  El último de los componentes de este modelo es el renderizador, este utiliza técnicas de renderizado volumétrico para generar una imagen final a partir de la nube de *voxeles* generada por la red neuronal.

4. Metodología

Para la realización del proyecto se va a aplicar una metodología ágil. Esto va a aportar gran flexibilidad en el desarrollo permitiendo la mejor adaptación a las necesidades o cambios que vayan surgiendo durante el mismo. La metodología ágil que se va a emplear va a ser Scrum, aunque con algunas modificaciones.

4.1 Scrum

La duración del proyecto tendrá un tiempo fijo, empezando el día 7 de diciembre de 2022 y acabando el día 13 de junio de 2023. Al establecer un tiempo fijo el alcance del proyecto será variable. Al principio, el desarrollo se centrará en cumplir los objetivos mínimos, una vez completados, se procederá a añadir mejoras y nuevas funcionalidades hasta la fecha de fin de proyecto.

El desarrollo se dividirá en sprints de 14 días laborables cada uno. En cada uno de estos sprints se harán labores tanto de desarrollo del proyecto a nivel de código como a nivel de documentación. Al final de cada sprint se va a obtener un producto entregable, ya sea en forma de documentación o en forma de código.

Entre el final de un sprint y el inicio del siguiente se dejará un margen aproximado de 2 días. Durante este margen se estudiarán los resultados del último sprint realizado, se subirá el progreso al repositorio de GitLab y se elegirán las tareas que se vayan a realizar en el siguiente sprint. Además, se podrán finalizar las tareas que hayan quedado incompletas del anterior sprint lo que aportará una flexibilidad extra al desarrollo del proyecto.

Los diferentes sprints se pueden dividir en tres fases según el avance del proyecto:

- La primera fase, conformada por los primeros sprints, se va a centrar en el estudio de todas las tecnologías y habilidades necesarias para la realización del proyecto. Esto incluye el aprendizaje de lenguajes de programación, el uso de librerías y frameworks, familiarización con entornos de desarrollo, etc. También se realizarán algunas pruebas para comprobar que todo funciona correctamente. Una vez acabados los estudios necesarios, se analizarán los requisitos del proyecto y se realizarán las primeras aproximaciones de los diseños de arquitectura e interfaces.

- La segunda fase se va a focalizar en la implementación y el cumplimiento de los objetivos básicos del proyecto. Esta fase es la más larga e importante ya que engloba todo el desarrollo del proyecto junto con la documentación. En los diferentes sprints se implementará todo el software que conforma el producto final, y, simultáneamente, se procederá a completar los diseños de arquitectura, datos e interfaces y el resto de la documentación necesaria.

- Una vez los objetivos básicos estén completados, comenzará la tercera fase del desarrollo. Esta consistirá en añadir nuevas funcionalidades y mejoras al proyecto de forma incremental. La duración de esta fase es muy variable y dependerá en gran medida del avance de todo el proyecto.

4.2 Backlog del producto

En la metodología Scrum se utiliza un documento denominado *backlog* del producto o *product backlog*. En él, se definen todas las tareas necesarias para el desarrollo de un proyecto software completo. Está formado por historias de usuario ordenadas por prioridad.

Las historias de usuario son descripciones de los requisitos hechas desde el punto de vista del cliente. En ellas se define un rol, una funcionalidad y el resultado esperado mediante el lenguaje natural. Para su definición se suele utilizar la plantilla: “Como [perfil], [quiero] [para]”. Por ejemplo: “*Como Vendedor, quiero registrar los productos y cantidades que me solicita un cliente para crear un pedido de venta”.*

Cada una de estas historias se dividen en tareas. Las tareas representan los pasos a seguir para cumplir cada una de las historias de usuario y están divididas de tal forma que su duración no sea de más de 2 o 3 días.

En caso de que una historia de usuario sea demasiado extensa, dure 10 días o más, se tiene que dividir en historias de usuario más pequeñas agrupadas en una épica de usuario.

El backlog del producto es un documento muy cambiante, al final de cada sprint se revisa y se modifica. Se pueden añadir o quitar historias de usuario o tareas, así como cambiar las prioridades, todo ello dependiendo del interés del cliente.

Al inicio de cada sprint se eligen las historias de usuario más prioritarias para su implementación por lo que estas tienen que estar definidas con un nivel de detalle superior. En cambio, las historias de usuario menos prioritarias no necesitan de tantos detalles y serán definidas más adelante, cuando se acerque la hora de implementarlas. Además, esto permite tener en cuenta todo el progreso del proyecto actual para poder definir y estimar mejor los tiempos de desarrollo de estas historias.

La siguiente tabla corresponde con el *product backlog* inicial:

Nº	Historia de usuario	Estado
1	Aprender a desarrollar aplicaciones Android	Siguiente (1)
2	Aprender manejo de datos complejo en apps Android	Siguiente (1)
3	Aprender a desarrollar aplicaciones con realidad aumentada	
4	Aprender el uso de Nerfstudio para la generación de modelos 3D	
5	Como usuario, quiero ver todos los objetos guardados para poder gestionarlos	
6	Como usuario, quiero tener la capacidad de ver todos los detalles de un objeto elegido para gestionarlo	
7	Como usuario, quiero tener la capacidad de eliminar los objetos guardados	
8	Como usuario, quiero tener la capacidad de modificar la información de los objetos guardados	

9	Como usuario, quiero tener la capacidad de visualizar un objeto en realidad aumentada	
10	Como usuario, quiero poder generar un objeto 3D utilizando un conjunto de imágenes para poder visualizarlo en realidad aumentada	
11	Como usuario, quiero poder generar un objeto 3D en base a un vídeo para posteriormente verlo en realidad aumentada	
12	Como usuario, quiero poder mover los objetos en el entorno de realidad aumentada	
13	Como usuario, quiero poder rotar los objetos dentro de la realidad aumentada	
14	Como usuario, quiero poder aumentar el tamaño del objeto dentro de la realidad aumentada	
15	Como usuario, me gustaría poder tomar el vídeo para la generación de objetos desde la propia app para no ocupar espacio en mi teléfono	
16	Como usuario, me gustaría poder tomar las fotos para la generación de objetos desde la propia app para no ocupar espacio en mi teléfono	
17	Como usuario, quiero poder iniciar sesión para que se guarden mis objetos	
18	Como usuario registrado, quiero poder modificar los datos de mi cuenta	
19	Como usuario registrado, quiero poder eliminar mi cuenta	
20	Como usuario, me gustaría elegir el tiempo que tarda en generarse un objeto	
21	Como usuario, me gustaría poder colocar varios objetos simultáneamente en realidad aumentada para probar distintas combinaciones	

El anexo II contiene el product backlog completo de una forma más detallada.

5. Tecnologías y herramientas

El proyecto estará desarrollado utilizando una arquitectura de tipo cliente-servidor. El lado del cliente consistirá la aplicación de Android, esta permitirá al usuario interactuar y visualizar la información almacenada en el servidor, así como visualizar los objetos en realidad aumentada. El lado del servidor estará formado por un servicio *REST* (Transferencia de Estado Representacional) junto con una base de datos y una *API* para la generación de modelos tridimensionales. A continuación, se describen las diferentes tecnologías que se van a utilizar para cada parte del proyecto.

5.1 Aplicación de móvil

Se va a implementar una aplicación para **Android** utilizando **Kotlin** como lenguaje de programación y **Android Studio** como IDE. La parte de realidad aumentada se implementará mediante el uso de la librería **SceneForm**.

Kotlin es el lenguaje oficial de Android. Es legible, fácil de aprender, seguro y estable, esto lo vuelve sencillo de usar y reduce el riesgo de errores y fallos en el código. Además, Kotlin es compatible con Java, lo que significa que permite la utilización de código existente escrito en este lenguaje.

Android Studio es un IDE especialmente diseñado para el desarrollo de aplicaciones Android. Es gratuito y de código abierto. Ofrece una interfaz intuitiva y fácil de usar, como se puede ver en la figura 15 está dotado de una serie de paneles y ventanas que permiten acceder a una amplia variedad de herramientas y recursos para crear aplicaciones de manera más eficiente y rápida. También integra un emulador de Android para la realización de pruebas en diferentes dispositivos y versiones de Android sin tener que usar dispositivos físicos.

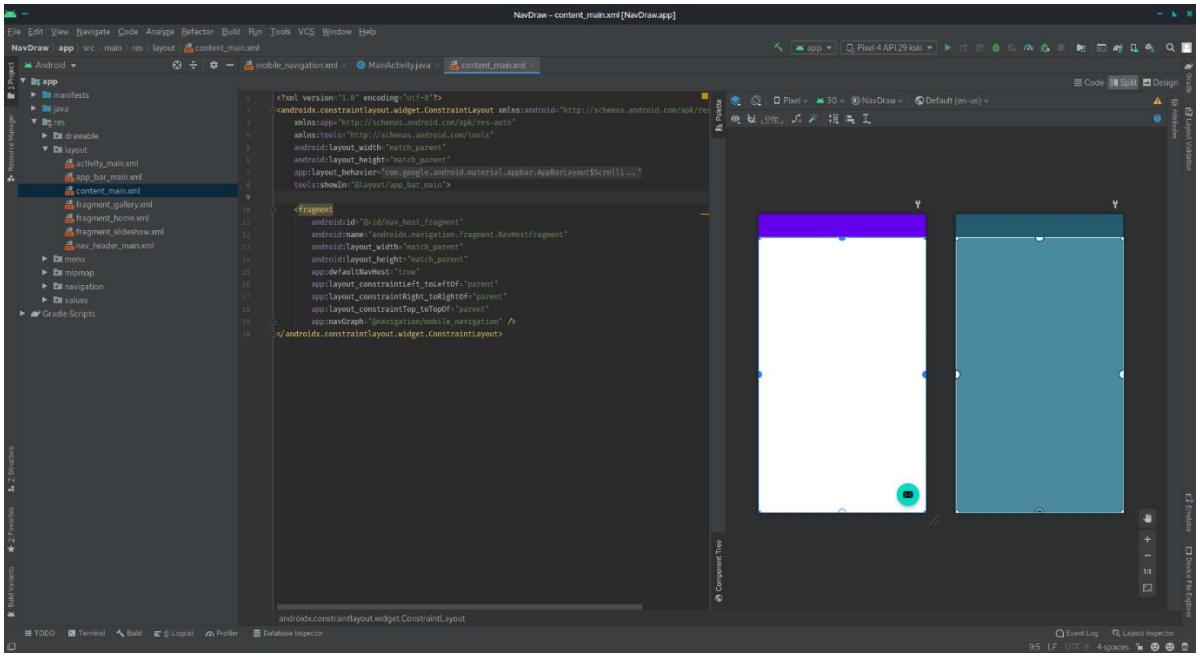


Figura 15 Android Studio

SceneForm es una librería que permite el desarrollo de aplicaciones de realidad aumentada para dispositivos móviles. Funciona mediante un sistema que analiza el entorno, genera un mapa virtual y permite colocar elementos tridimensionales en la escena. Es fácil de integrar y es compatible con una amplia gama de dispositivos Android.

5.2 Servicio REST

El servicio *REST* se va a implementar utilizando el *framework* de ***Spring Boot*** con el lenguaje **Kotlin.** Este servicio estará conectado a una base de datos **PostgreSQL** para la persistencia de los datos. Como IDE se utilizará **IntelliJ IDEA**. Tanto el servicio *REST* como la base de datos se van a desplegar dentro de contenedores de **Docker**. Para realizar las distintas pruebas con el servidor se va a utilizar la herramienta **Postman**.

Spring Boot es un *framework* que permite el desarrollo y el despliegue de servicios *REST*. Esta tecnología se encarga de gestionar las dependencias y el despliegue del servicio de aplicaciones permitiendo a los desarrolladores centrarse en el desarrollo. Utiliza un servidor de aplicaciones embebido *Tomcat* para levantar el servicio. Para la creación de un proyecto con *Spring Boot* es tan sencillo como entrar en la página de *Spring Initializr*[11], elegir las dependencias y pulsar el botón de generar. Esto descarga un fichero comprimido que se puede compilar y ejecutar directamente para levantar un servidor *REST*.

IntelliJ IDEA es un IDE para el desarrollo de software. Es ampliamente utilizado junto con lenguajes como Java y Kotlin. Además, dispone de una versión gratuita equipada con bastantes herramientas que facilitan el desarrollo y las pruebas de productos software.

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto, es fácil de instalar e integrar con *Spring Boot*. También es ligero, eficiente y muy popular.

Docker es una plataforma que facilita el despliegue de forma rápida de distintos tipos de productos software dentro de contenedores. Un contenedor es un paquete de *software* aislado que contiene todos los elementos necesarios para levantar servicios WEB, bases de datos o incluso sistemas operativos de una forma sencilla. Estos contenedores son similares a las máquinas virtuales, pero son mucho más ligeros ya que solo contienen aquellos elementos imprescindibles para su funcionamiento. Además, el uso de contenedores garantiza el funcionamiento del *software* independientemente del sistema operativo sobre el que se estén ejecutando. Esta herramienta se va a utilizar para levantar la base de datos, el servicio *REST* y la *API* de generación de objetos tridimensionales.

Postman es una herramienta que permite lanzar peticiones http para probar, entre otros, servicios *REST*. Es sencillo de usar y permite la creación de colecciones para una mejor organización.

5.3 API de generación de objetos 3D

Para el desarrollo de la *API* se utilizará el lenguaje **Python** junto con la librería **Nerfstudio**. Como IDE se utilizará **Visual Studio Code** y el despliegue de la *API* se realizará en un contenedor **Docker**.

Python es el principal lenguaje utilizado en el campo de la Inteligencia artificial además de ser potente y sencillo.

Nerfstudio es una librería pública que proporciona varias versiones de modelos de inteligencia artificial *NeRF* para la generación de objetos tridimensionales de forma sencilla. También tiene varias funciones que permiten exportar los objetos generados en forma de nubes de puntos o de mallas de triángulos.

Visual Studio Code es un editor de código abierto multiplataforma, muy potente, flexible y ligero. Además, cuenta con la posibilidad de añadir extensiones para personalizar la experiencia de desarrollo.

5.4 Metodología

Para aplicar la metodología scrum se va a utilizar **YouTrack**[12]. Esta herramienta permite la creación de tableros scrum con múltiples columnas que representan el estado de las diferentes tarjetas como se puede ver en la figura 16. Cada tarjeta representa una historia de usuario o tarea a desarrollar e incluye: un resumen, una descripción y un tiempo estimado. YouTrack también permite la creación y organización del proyecto en sprints de desarrollo y facilita la gestión de los mismos ya que es capaz de generar informes de forma automática.

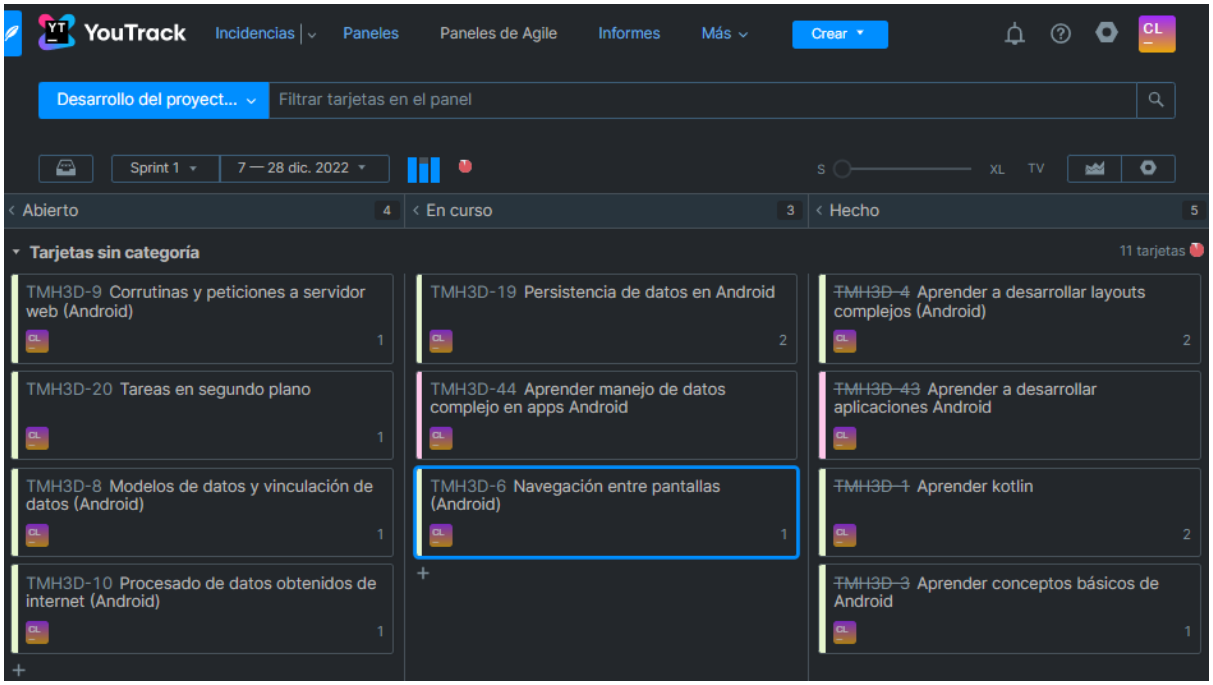


Figura 16 YouTrack

5.4 Controlador de versiones

Como repositorio se va a utilizar **GitLab**[13], esta plataforma proporciona un lugar centralizado para almacenar y gestionar el código fuente del proyecto. Utiliza la tecnología **Git** para el control de versiones, lo que facilita en gran medida el seguimiento y la gestión de todas las versiones del proyecto a lo largo de su desarrollo.

6. Aspectos relevantes del desarrollo

6.1 Política de ramificación de Git

Para el control de versiones se van a utilizar dos ramas principales que van a estar presentes durante todo el desarrollo del proyecto, estas son *main* y *develop*. Además de estas ramas, se van a utilizar una serie de ramas de apoyo, las ramas *feature*[14].

La rama ***main o master*** es la principal, en ella se encontrarán las diferentes versiones de producción del proyecto. Estas versiones serán estables y deberán tener un *tag* que contenga cierta información de las mismas. De esta rama nace la rama *develop* como se ve en la figura 17.

En la rama ***develop*** es donde se irán realizando todos los cambios del proyecto, esta representa el avance del mismo y contiene los últimos cambios realizados. Cuando el código está lo suficientemente desarrollado y es estable, esta rama se mezcla con la rama *main* dando origen a una nueva versión de producción.

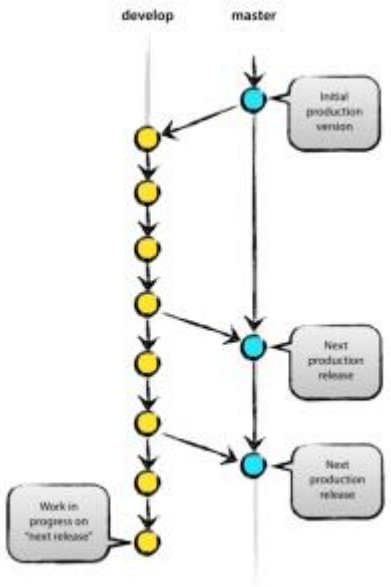
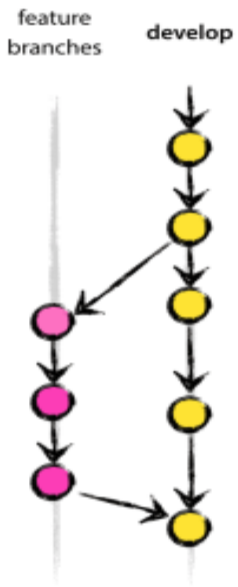


Figura 17 Ramas principales Git [14]



Por otro lado, las **ramas *feature*** derivan de la rama *develop*. Se van a utilizar para desarrollar nuevas características, arreglar errores y preparar el código para lanzar las versiones de producción. Estas ramas se crean al inicio de cualquiera de las anteriores actividades y solo existen durante estas. Al finalizar, se mezclarán de vuelta con *develop* o se descartarán dependiendo de su éxito.

Todas las ramas de este tipo tendrán un nombre que identifique la actividad que se realiza, este será: feature-[descripción de la actividad] En la figura 18 se puede ver un ejemplo de estas ramas.

Figura 18 Ramas feature[14]

6.2 Desarrollo del proyecto

6.2.1 Aprendizaje previo

El primer sprint se ha enfocado en el aprendizaje de los conceptos necesarios para el desarrollo de aplicaciones Android.

Se ha realizado el curso gratuito **Aspectos Básicos de Android en Kotlin** [15] de la página de Android para desarrolladores. Este curso está divido en varios temas que engloban desde aprender los conceptos básicos de Kotlin como lenguaje de programación hasta aprender a manejar bases de datos y conectarse a través de internet.

Cabe destacar el aprendizaje del desarrollo de las distintas pantallas de la aplicación y la implementación de diferentes funcionalidades de los elementos en pantalla como botones o listas desplazables. Esto va a ser importante a lo largo del proyecto ya que va a ser la forma de presentarle toda la información y las utilidades de la aplicación al usuario final. Algunos ejemplos de este tipo de interfaces se pueden ver en las figuras 19 y 20. Una aplicación con diferentes elementos visuales como cuadros de selección y botones a la izquierda y otra con una lista desplazable de objetos a la derecha.

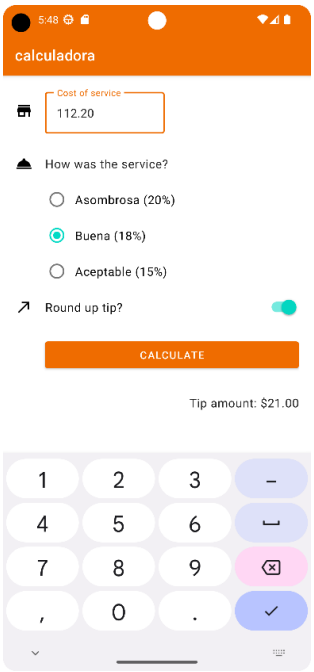


Figura 19 Aplicación Calculadora

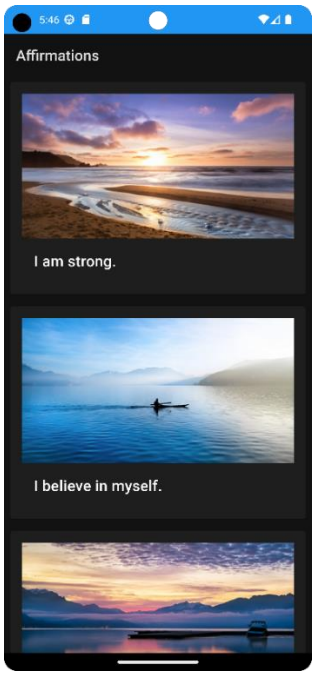


Figura 20 Aplicación Afirmaciones

Otra de las cosas a destacar del curso es la navegación entre pantallas. Esta es una de las claves a la hora de desarrollar aplicaciones ya que define el modo en el que el usuario puede moverse por la aplicación. También hay que mencionar la importancia de guardar los datos en modelos de forma que se permita compartir información entre diferentes interfaces. Una mala implementación podría resultar en una app difícil de usar, lenta o ineficiente. En Android, la navegación entre pantallas se puede desarrollar utilizando herramientas como la de la figura 21, donde las flechas indican las posibles transiciones entre pantallas.

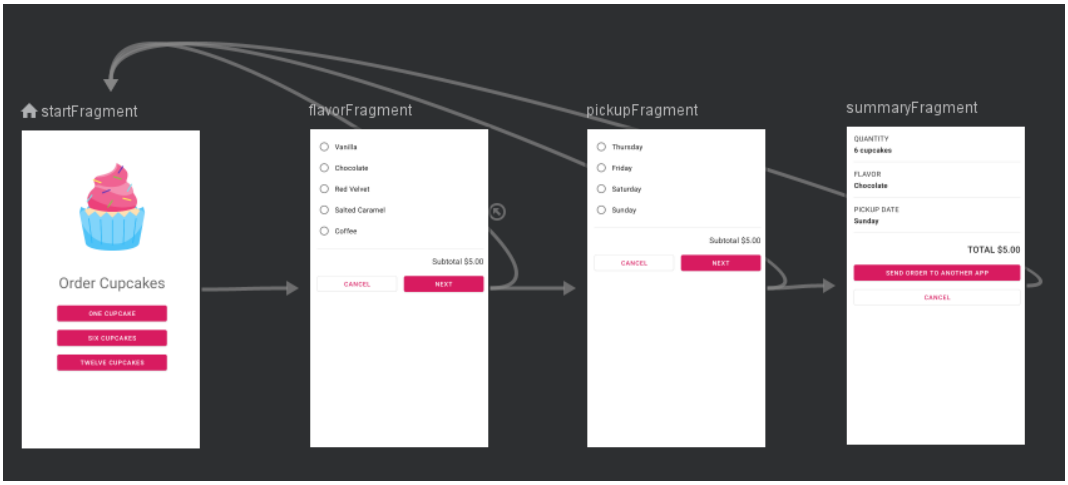


Figura 21 Aplicación Magdalenas

Otro de los aspectos relevantes del desarrollo de aplicaciones es la conexión a Internet. La aplicación que se va a desarrollar en este proyecto utilizará solicitudes a servidores web para recuperar la información almacenada en el lado del servidor. Se van a utilizar las librerías Retrofit[16] y Moshi[17] ya que proporcionan una forma mucho más sencilla y rápida de realizar estas solicitudes y de interpretar la información recibida. Además, como los modelos 3D y las imágenes almacenadas son elementos que pesan y pueden tardar bastante en descargarse, se va a utilizar la librería Coil[18], esta permite la descarga en segundo plano de una forma óptima y sencilla. En la figura 22 se puede ver un ejemplo de una aplicación que utiliza Coil para recuperar imágenes de un servidor, mientras que en la figura 23 se puede ver otro ejemplo de app que recupera distintos tipos de información de un servidor web.

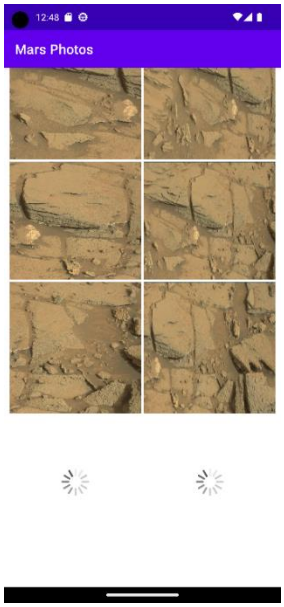


Figura 22 Aplicación Fotos de marte

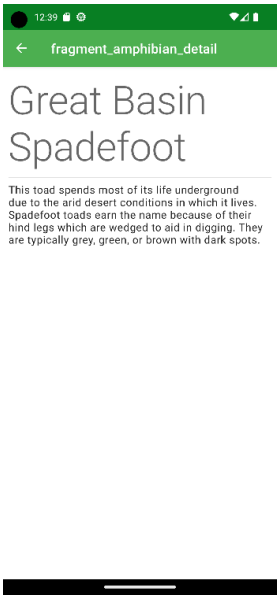


Figura 23 Aplicación Anfibios

6.2.2 Primeras pruebas

El segundo sprint se ha centrado en probar las partes del proyecto relacionadas con la inteligencia artificial y la realidad aumentada.

Para la realización del proyecto se va a utilizar la librería Nerfstudio, esta proporciona varias versiones de modelos *NeRF* para la generación de objetos en tres dimensiones. Incluye varios métodos para procesar los datos ya que requiere un formato específico de entrada. También tiene varias funciones que permiten exportar los objetos generados en forma de nubes de puntos o de mallas de triángulos.

La instalación de Nerfstudio requiere la instalación previa de varias librerías y drivers en el dispositivo ya que hace uso de la GPU para el entrenamiento de los modelos. Esto, unido con la cantidad de versiones de los diferentes productos de software necesarios, hace que la instalación sea bastante compleja ya que se provocan muchas incompatibilidades y errores. Tras varios intentos de instalación fallidos, finalmente, se ha realizado la instalación en un contenedor de Docker haciendo uso de la imagen oficial de Nerfstudio y añadiendo todas las librerías de Python necesarias para el desarrollo de la *API*.

Se han realizado varias pruebas de generación de modelos tridimensionales utilizando los datos de prueba que proporciona la propia librería. Se puede visualizar todo el proceso en tiempo real ya que Nerfstudio proporciona un modo espectador como el de la figura 24. En esta imagen se puede ver el entorno tridimensional generado por el modelo junto con las imágenes utilizadas para su entrenamiento, estas están colocadas en las posiciones en las que fueron tomadas.

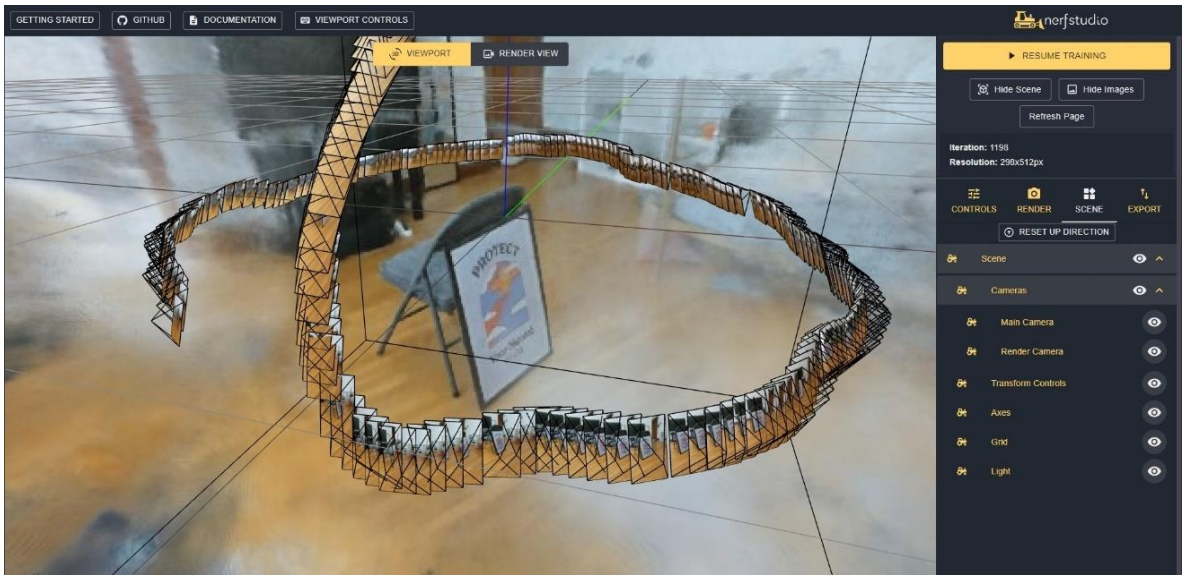


Figura 24 Nerfstudio viewer

Los resultados se van guardando a medida que se entrena el modelo. Utilizando estos resultados almacenados se puede exportar el objeto en forma de malla. Una vez se han realizado las transformaciones necesarias se obtienen varios archivos, entre ellos un .obj que contiene los polígonos que forman el modelo y un .png con las texturas.

Para probar la realidad aumentada se ha descargado la aplicación de ejemplo que proporciona Google. La aplicación escanea el entorno en busca de superficies planas y, una vez detectadas permite colocar objetos 3D tocando la pantalla.

Como se observa en la figura 25, aparece una especie de malla sobre las superficies detectadas. Al pulsar en alguna zona de la malla se posiciona el objeto, se puede cambiar la posición de la cámara y el objeto mantendrá la posición relativa al mundo real.

Figura 25 Realidad aumentada ARCore

Figura 26 Silla en realidad aumentada, ARCore

El siguiente paso ha sido cambiar el objeto que se visualiza en la aplicación por el generado anteriormente con Nerfstudio para comprobar que todo funciona correctamente. Los resultados del experimento se pueden ver en la figura 26.

En la imagen aparece una silla. Aunque los resultados no son especialmente buenos, estos nos proporcionan bastante información relevante.

Para empezar, el objeto tiene una rotación errónea que tendrá que ser corregida de alguna manera.

Otro de los problemas es el suelo, los modelos *NeRF* son capaces de reconstruir habitaciones enteras. En este caso, al recortar el objeto, una parte del suelo no ha sido eliminada haciendo que el objeto se distinga mucho peor.

6.2.3 Gestión de objetos

Se ha comenzado el desarrollo tanto de la parte del servidor como de la aplicación de móvil de forma simultánea.

Se ha definido una entidad llamada *ObjetoData* que representa toda la información relacionada con cada uno de los objetos 3D que maneja la app, esto va a permitir una fácil gestión de los objetos por parte del usuario. Esta entidad contiene los siguientes campos:

- Id (valor único con el que el objeto se almacena en la base de datos)
- Nombre
- Descripción
- Fecha (fecha de creación del objeto)
- Peso (espacio que ocupa el objeto)
- Estado (indica si el objeto se puede visualizar en RA)

En la parte del servidor se han definido una serie de *endpoints* o rutas. Estas rutas actúan a modo de interfaz con el exterior, cuando el servidor recibe una petición *http* en alguna de ellas, ejecutará una serie de acciones y devolverá una respuesta.

En este caso, los *endpoints* van a permitir la modificación, publicación y eliminación de los objetos previamente definidos. El servidor atenderá las peticiones *http* a estos *endpoints* modificando la información almacenada en la base de datos y devolviendo los datos solicitados cuando sea necesario. La siguiente tabla contiene las rutas desarrolladas a lo largo de esta fase.

Método	URI	Descripción
Get	/objeto/info/{id}	Devuelve el objeto cuyo id coincide con el enviado en la solicitud.
Get	/objeto/all	Devuelve una lista con todos los objetos almacenados en la base de datos.
Put	/objeto/{id}	Modifica el objeto por id. Requiere el envío de un objeto nuevo en el cuerpo de la petición y devuelve el objeto una vez modificado.
Post	/objeto	Permite la creación de un nuevo objeto en la base de datos. Requiere el envío en un objeto en el cuerpo de la petición.
Delete	/objeto/{id}	Elimina un objeto según su id.

En la parte de Android se ha implementado la primera versión de la interfaz gráfica. Todas las pantallas desarrolladas junto con la navegación se pueden observar en la figura 27.

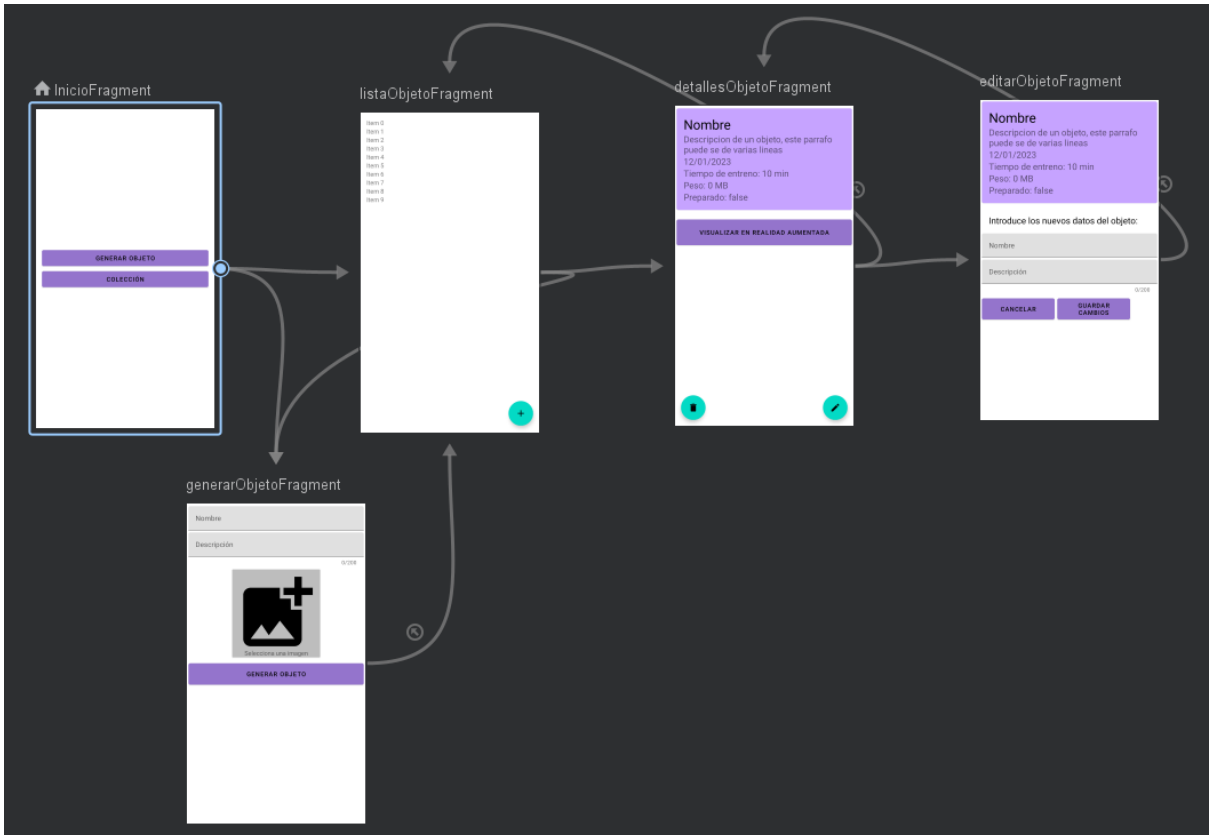


Figura 27 Primera aproximación a la interfaz gráfica

La primera de las pantallas es un menú principal que permite elegir entre varios caminos posibles. Uno de estos caminos es la colección o lista de objetos, esta muestra una lista con todos los objetos que han sido recuperados del servidor.

Al seleccionar uno de los objetos, la app muestra su información detallada y permite al usuario su eliminación o modificación. Por último, la pantalla que aparece en la parte inferior se corresponde con la creación de objetos. Tanto la creación como la modificación de los objetos se realiza mediante formularios. Todas las acciones que se realizan en la app realizan llamadas http para registrar los cambios en el lado del servidor.

6.2.4 Realidad aumentada

Inicialmente, la realidad aumentada iba a ser desarrollada haciendo uso de la librería ARCore. Aunque esta librería realiza todas las acciones necesarias para escanear los planos y hacer el seguimiento de los objetos en la escena, no viene con herramientas para el renderizado de objetos tridimensionales. Esto es un problema ya que se necesita un elevado conocimiento de *openGL*, una *API* de renderizado de gráficos, para poder realizar esa tarea.

Por ello, se ha optado por utilizar una librería llamada SceneForm[19], esta librería nació como un proyecto de Google con el objetivo de acercar la realidad aumentada a los desarrolladores de aplicaciones móviles sin necesidad de amplios conocimientos de *openGL*. Este proyecto de código abierto se archivó por parte de la compañía en el año 2020, pero una serie de usuarios de GitHub han continuado desarrollando y manteniendo la librería.

Mediante el uso de SceneForm se ha desarrollado una nueva pantalla en la aplicación móvil que puede ser accedida desde el apartado de detalles de un objeto donde se pueden visualizar los modelos 3D en realidad aumentada tal como se observa en las figuras 28 y 29.

Figura 28 Cojín en realidad aumentada

Figura 29 Controles realidad aumentada

Al abrir esta pantalla, SceneForm se encarga de crear una sesión de realidad aumentada. Utiliza el software de ARCore para la detección de los planos y el seguimiento de los objetos mientras que, por otro lado, se encarga del renderizado de los objetos tridimensionales.

Cuando el usuario toca uno de los planos detectados, la librería devuelve las coordenadas del punto permitiendo crear un anclaje. Los anclajes son un tipo de datos que representan la posición de un objeto con respecto a un plano, en este caso, el plano con el que se ha interactuado, esto permite hacer un seguimiento de los objetos.

Adicionalmente, se han desarrollado una serie de controles como los que se ven en la figura 29 para poder interactuar con el objeto de la escena. Haciendo uso de los botones y los controles deslizantes se puede alterar tanto la rotación como el tamaño del objeto. También se puede restablecer la rotación y tamaño originales y eliminar el objeto de la escena. La propia pantalla dispone de información con explicaciones sobre estas funcionalidades.

Para cambiar la posición del objeto solo es necesario tocar de nuevo en uno de los planos detectados, esto eliminará el anclaje creado anteriormente y creará uno nuevo usando las nuevas coordenadas devueltas por la librería.

Para gestionar los modelos tridimensionales se han desarrollado varios *endpoints* en el lado del servidor que permiten almacenar y extraer los archivos necesarios de la base de datos.

6.2.5 Generación de modelos 3D

Como se explicó al principio del documento, se ha desarrollado un *API* para la generación de los modelos tridimensionales. Esta *API* cuenta con dos *endpoints,* uno para la generación con imágenes y otro para la generación con vídeo. El proceso de generación de objetos sigue los pasos que aparecen en la figura 30.

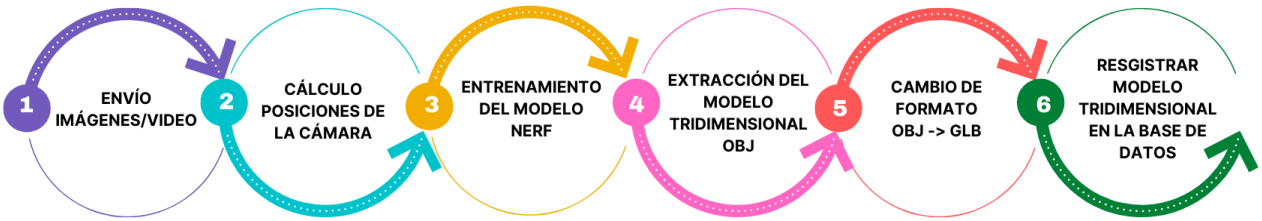


Figura 30 Generación de modelos tridimensionales

1. Envío de las imágenes/vídeo a la API.

En este primer paso, el dispositivo Android envía las imágenes o el vídeo para la generación del modelo 3D a la *API*. En caso de imágenes, estas son guardadas en una carpeta y se inicia el siguiente paso.

En caso de vídeo es necesario un paso adicional para dividirlo en fotogramas. La librería de Nerfstudio cuenta con una función que realiza este paso, pero con un inconveniente: se extraen todos los fotogramas del vídeo, esto ralentiza el resto de los pasos de forma considerable. Por lo tanto, se ha optado por otra alternativa.

Se van a extraer un número limitado de imágenes. Con el fin de obtener la mayor cantidad de perspectivas del objeto o escena, se eligen de tal forma que estén repartidas a lo largo de todo el vídeo. Para lograr este objetivo, se divide el número de fotogramas del vídeo entre un valor preestablecido n, el resultado simboliza la cantidad de imágenes que se van a omitir entre cada extracción. Por ejemplo, si el número de fotogramas es de 500, y el valor de n es de 100, el resultado daría 5, por lo que se extraerían 1 de cada 5 imágenes, tal y como se muestra en la figura 31.

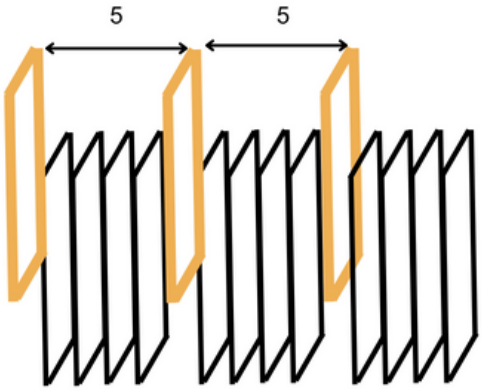


Figura 31 Extracción de imágenes

2. Cálculo de las posiciones de la cámara.

Este paso consiste en extraer las posiciones (x, y, z) y las direcciones (θ, ϕ) de las imágenes enviadas al servidor. El modelo de inteligencia artificial hace uso de esta información a la hora de hacer la reconstrucción de la escena por lo que es totalmente imprescindible.

En la figura 32 se puede ver un esquema de este paso donde se utiliza una función de la librería Nerfstudio que, mediante un componente llamado *Colmap*, es capaz de calcular las posiciones (x, y, z) y las direcciones (θ, ϕ) de cada una de las imágenes. La información extraída se almacena en un fichero que posteriormente será usado para el entrenamiento del modelo.

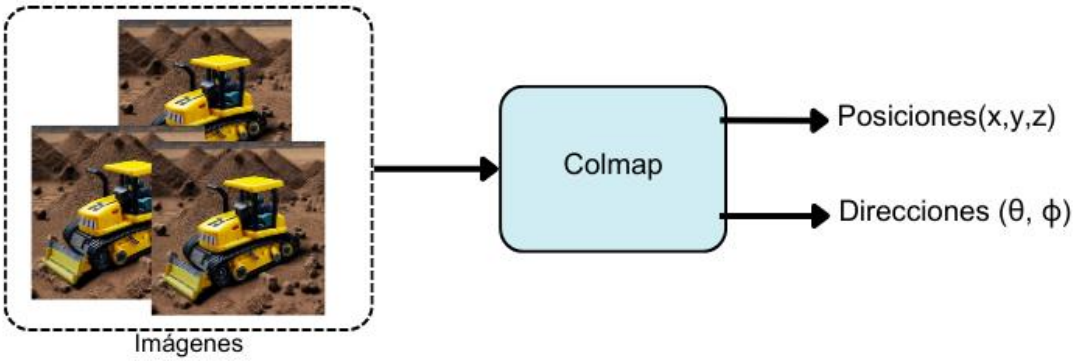


Figura 32 Extracción de posiciones y direcciones de las imágenes

Para calcular las posiciones y ángulos de la cámara, las imágenes recibidas deben tener un cierto nivel de solapamiento, es decir, que cada elemento aparezca en más de una imagen. En caso de que las imágenes no cumplan este requisito el proceso puede llegar a fallar. En caso de fallo, se envía una petición al servidor para que quede registrado en la base de datos.

Existen métodos que permiten extraer esta información en el momento en que se capturan las imágenes haciendo posible saltarse este paso. El problema de esta alternativa es que solo unos pocos dispositivos tienen tecnología suficiente para realizarla. Por ese motivo, en este proyecto se ha optado por el cálculo de las posiciones en el lado del servidor.

3. Entrenamiento del modelo NeRF.

En la figura 33 se puede ver el esquema de este paso donde se hace uso del modelo Nerfacto descrito en la sección 3.4.1. Este modelo utiliza las posiciones y direcciones calculadas en el paso anterior junto con las imágenes enviadas al servidor para su entrenamiento.

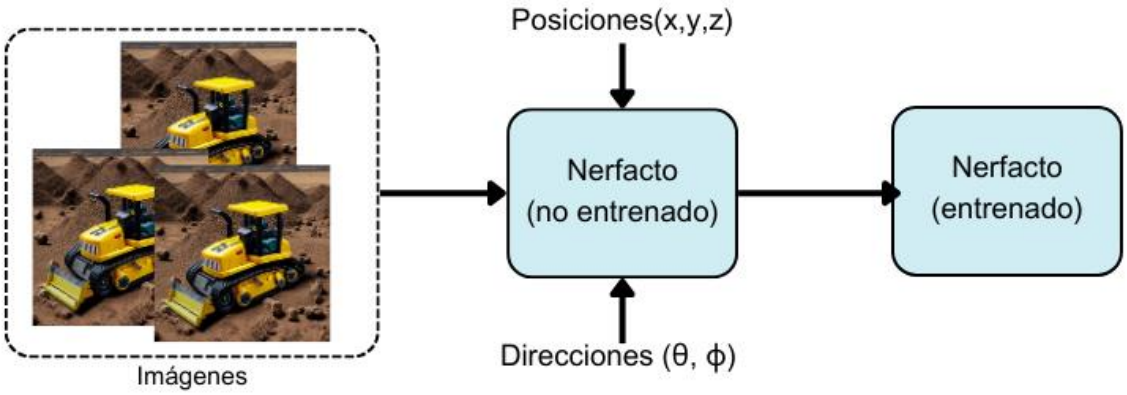


Figura 33 Entrenamiento del modelo Nerfacto

El proceso de entrenamiento consiste en optimizar los parámetros de la red neuronal interna del propio modelo Nerfacto siguiendo el proceso descrito en el punto 3.4. Esta red interna será la encargada de aprender a interpretar la escena tridimensional capturada.

Se puede considerar que los pesos de la red contienen la información tridimensional de la escena. Al principio, los pesos tomarán valores aleatorios, esto significa que el modelo tendrá internamente una representación del objeto muy alejada de la escena real capturada. A medida que el entrenamiento de la red neuronal avance, los pesos convergerán en valores más cercanos a los óptimos y por lo tanto el modelo contará con una representación más cercana a la escena original.

El proceso de entrenamiento tiene una duración variable que depende tanto de la potencia de la GPU como del número de pasos en el entrenamiento. El número de pasos simboliza la cantidad de iteraciones del modelo, a mayor cantidad se lograrán mejores resultados a cambio de un mayor tiempo de procesamiento. Para evitar tiempos muy largos y debido a las limitaciones de *hardware* se ha establecido el número de pasos en 3000, consiguiendo un tiempo medio de alrededor de 15 minutos.

En la figura 34 se puede ver un ejemplo de ejecución del entrenamiento del modelo Nerfacto. En la imagen se observan varias filas de información sobre el entrenamiento. En cada fila aparece el número de pasos ejecutados, el tiempo medio de cada uno de los pasos, la estimación del tiempo restante y el número medio de rayos procesados por segundo. En este ejemplo se han ejecutado un total de 1000 iteraciones en alrededor de 5 minutos.

```
Step (% Done)       Train Iter (time)    ETA (time)           Train Rays / Sec
--------------------------------------------------------------------------------
890 (89.00%)        298.772 ms           32 s, 864.890 ms     14.03 K
900 (90.00%)        299.543 ms           29 s, 954.278 ms     14.00 K
Step (% Done)
--------------------
910 (91.00%)        299.684 ms           26 s, 971.588 ms     13.99 K
920 (92.00%)        299.634 ms           23 s, 970.749 ms     14.00 K
930 (93.00%)        300.347 ms           21 s, 24.283 ms      13.97 K
940 (94.00%)        301.564 ms           18 s, 93.837 ms      13.90 K
950 (95.00%)        304.246 ms           15 s, 212.303 ms     13.76 K
960 (96.00%)        305.220 ms           12 s, 208.796 ms     13.72 K
970 (97.00%)        304.341 ms           9 s, 130.231 ms      13.76 K
980 (98.00%)        306.581 ms           6 s, 131.626 ms      13.67 K
990 (99.00%)        305.991 ms           3 s, 59.911 ms       13.70 K
999 (99.90%)
--------------------------------------------------------------------------------
```

Figura 34 Información del entrenamiento del modelo Nerfacto

4. Extracción del modelo tridimensional del modelo NeRF.

Una vez el entrenamiento ha finalizado, el modelo habrá aprendido a predecir los colores y densidades de los voxeles del objeto tridimensional. A partir de ahí, se puede usar tanto para predecir imágenes desde nuevas perspectivas como para predecir la nube de puntos que conforman el objeto.

Para extraer el modelo tridimensional se utiliza otra de las funciones de la librería, que se va a encargar de hacer todas las tareas necesarias para guardar el objeto resultado.

- Primero, genera una nube de puntos a partir del estado actual de la red neuronal.
- A partir de la nube de puntos, genera una malla de triángulos, esta representa la forma del objeto o escena capturados.
- La malla de triángulos se desenvuelve para generar un mapa de dos dimensiones con las texturas del objeto. Las texturas representan los colores y tonalidades del objeto.
- Los resultados son almacenados en varios ficheros: un fichero .obj con los polígonos del objeto y una imagen en formato .png con las texturas generadas.

En la figura 35 se puede ver el esquema de este proceso.

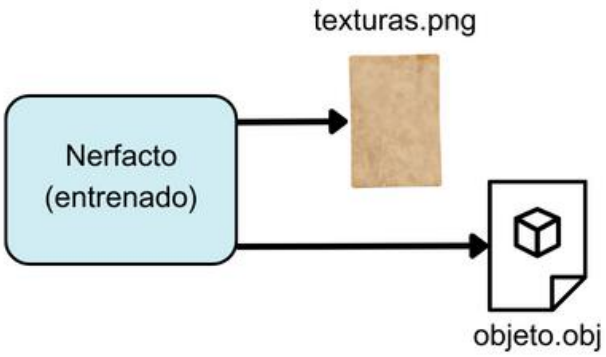


Figura 35 Extracción del modelo tridimensional

Teniendo en cuenta el uso de una tarjeta gráfica NVIDIA GTX 1060, este proceso lleva alrededor de 20 minutos consiguiendo unos resultados bastante aceptables. Algunos de los resultados obtenidos son los que aparecen en las figuras 36 y 37.

Figura 36 Modelo 3D: cojín y mesa

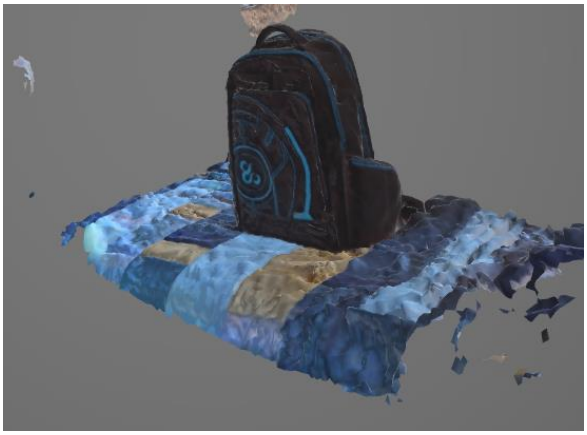


Figura 37 Modelo 3D: mochila

Se puede observar cómo los diferentes objetos capturados no están completamente aislados, este problema es bastante difícil de resolver de forma automática debido a la multitud de posibles tamaños de los objetos escaneados. La calidad final de los modelos tridimensionales generados depende en gran medida de las imágenes capturadas, llegando a producir, en algunos casos, resultados con muy poca calidad.

5. Cambio de formato.

Para la visualización en realidad aumentada de los modelos tridimensionales se usa una librería llamada *SceneForm*, esta librería requiere que los objetos que se quieran visualizar estén en formato binario *glTF*, por lo que es necesario un cambio de formato.

Para realizar el cambio de formato se ha utilizado un script[20] que, mediante el uso de *Blender*, importa los ficheros generados en el paso anterior y exporta el objeto en el formato correcto como se observa en la figura 38.

Blender es una herramienta de código abierto para el modelado y renderizado de gráficos 3D, se usa principalmente para crear películas animadas, efectos visuales y modelos de impresión 3D.

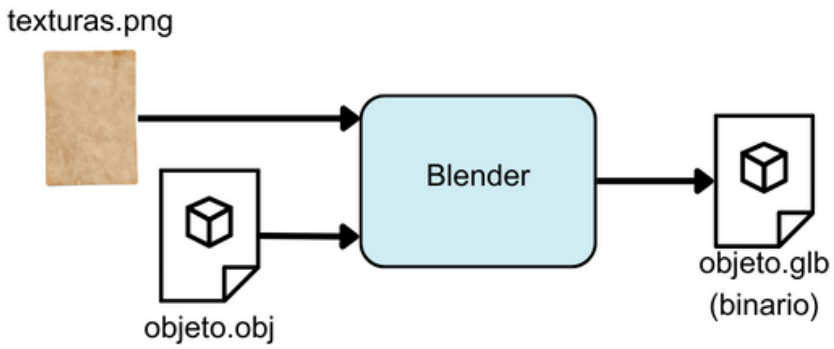


Figura 38 Cambio de formato

6. Registro del modelo 3D en la base de datos.

Para finalizar el proceso, el fichero *glTF* generado se envía al servicio *REST* para ser almacenado en la base de datos. Una vez se completa el proceso, el usuario tendrá la posibilidad de descargar el modelo tridimensional en su dispositivo móvil para su visualización en realidad aumentada.

6.2.6 Captura de imagen y vídeo

Para la captura de imagen y vídeo desde la aplicación se ha utilizado una librería que ofrece el propio Android llamada *CameraX*[21]. Esta librería permite configurar la cámara del dispositivo de forma sencilla y con una gran compatibilidad entre diferentes dispositivos.

Se ha desarrollado una pantalla que muestra una previsualización de la imagen de la cámara. En función de la elección previa del usuario, se podrán capturar varias imágenes como se observa en la figura 39 o un vídeo como se ve en la figura 40.

En caso de imágenes, se muestra una vista previa en la parte inferior izquierda con la previsualización de la última imagen capturada junto con un conteo del número de imágenes en la parte derecha.

Por otro lado, en caso de vídeo, se muestra el total de segundos grabados. El usuario podrá pausar y reanudar la grabación las veces que quiera, así como detenerla y volver a grabar en caso de no estar conforme con el resultado.

Tanto las imágenes como los vídeos se guardan de forma temporal en la memoria interna de la aplicación hasta que se hayan terminado de enviar a la *API*. Estos ficheros no pueden ser accedidos desde fuera de la app y se borran para no ocupar espacio.

Figura 39 Pantalla cámara para fotos

Figura 40 Pantalla cámara para vídeo

6.2.7 Aspecto final de la aplicación

Se han añadido algunas funcionalidades adicionales a la aplicación que se van a ir explicando en este apartado. También se han realizado mejoras de la interfaz para garantizar una mejor experiencia de usuario.

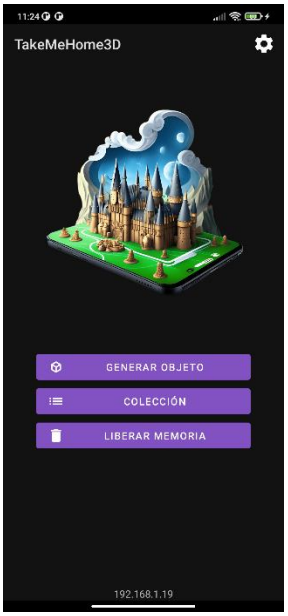


Figura 41 Pantalla de inicio

Nada más abrir la aplicación se va a mostrar una pantalla como la de la figura 41.

Esta es la pantalla de inicio, que funciona a modo de menú principal. Permite navegar a la colección de objetos o a la pantalla de generación de un nuevo objeto.

También incluye un botón para liberar la memoria, es decir, eliminar los distintos archivos temporales que la aplicación haya ido generando. Entre estos archivos se encuentran los modelos tridimensionales descargados en el dispositivo, que quedan almacenados para no tener que descargarse cada vez que se quiera acceder a ellos.

La generación de objetos sigue una serie de pasos que el usuario tiene que ir completando.

Primero, el usuario deberá introducir los datos del objeto a generar: nombre y descripción.

Luego, tendrá la opción de elegir entre utilizar imágenes o vídeo para la generación del modelo tridimensional.

Por último, tendrá que seleccionar las imágenes o el vídeo que desee utilizar. Para este último paso, el usuario tiene la opción de elegir el contenido multimedia desde la galería del móvil o, por el contrario, utilizar la cámara del dispositivo para capturar los datos necesarios como se explica en el punto 6.2.6.

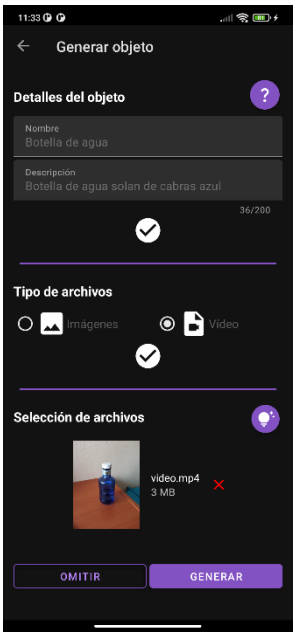


Figura 42 Generar con vídeo

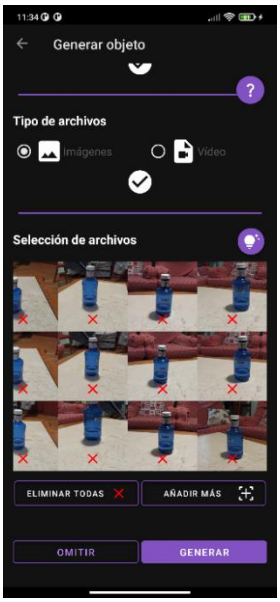


Figura 43 Generar con imágenes

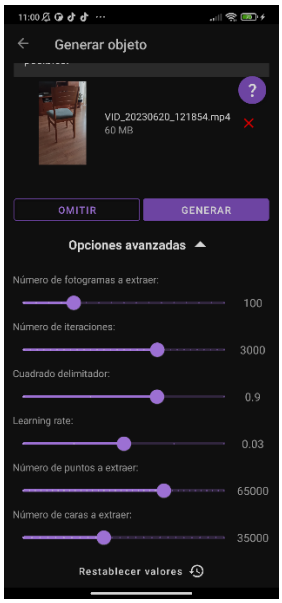


Figura 44 Parámetros del modelo

En la figura 42 se puede ver el proceso en caso utilizar un vídeo mientras que en la figura 43 se observa el mismo procedimiento para imágenes.

En ambos casos se da la posibilidad de deseleccionar uno o varios elementos o eliminarlos todos y repetir el proceso de selección.

Una vez se hayan completado todos los pasos el usuario podrá pulsar el botón de "generar" para que se envíe el contenido multimedia a la *API* y comience el proceso de generación.

En caso de que el usuario quiera dejar este paso para más adelante, tiene la opción de omitirlo, pudiendo realizarlo más tarde desde el apartado de detalles del objeto.

Adicionalmente, el usuario tiene la posibilidad de configurar varios parámetros de la generación de objetos 3D desde el desplegable "Opciones avanzadas" que aparece en la figura 44. Modificando los valores que aparecen en pantalla se puede conseguir un tiempo menor en la generación o una mejor calidad de resultados.

La colección de objetos se puede ver en la figura 45. Esta contiene una lista con la imagen y cierta información de cada objeto para que se puedan identificar de forma sencilla.

Cada elemento de la lista contiene el nombre, la fecha y el estado actual del objeto. El estado representa si el modelo tridimensional está preparado, en curso, pendiente o si el proceso de generación ha fallado.

La lista de elementos se recupera del servidor cada vez que se inicia la aplicación, pero también puede ser actualizada haciendo el gesto de deslizar hacia abajo sobre la pantalla.

Mediante la opción situada en la parte superior izquierda de la pantalla, se pueden ordenar los objetos en función de nombre o fecha de creación.

Cualquier objeto puede ser seleccionado para consultar sus detalles.

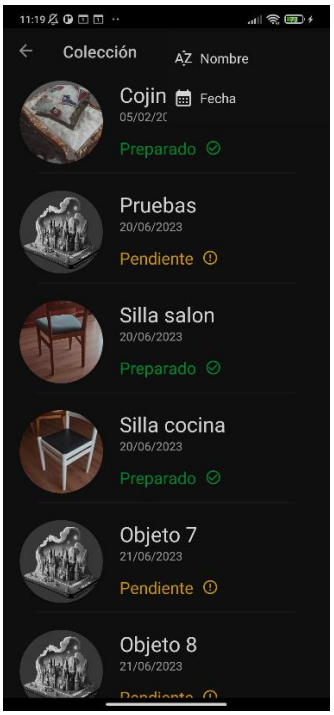


Figura 45 Lista de objetos

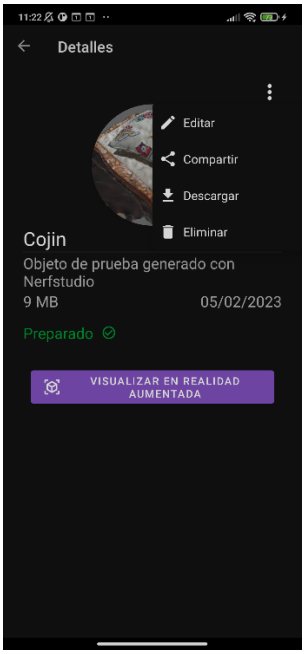


Figura 46 Detalles objeto

La pantalla de detalles se muestra en la figura 46. Como se puede observar, esta incluye información adicional como el peso en memoria del modelo tridimensional.

Además, incluye una serie de opciones que permiten editar, compartir, descargar y eliminar el objeto al pulsar sobre el icono de los tres puntos que aparece al lado del nombre.

El botón que aparece en mitad de la pantalla permite al usuario visualizar el objeto en realidad aumentada tal y como se explica en el punto 6.2.4.

En caso de que el objeto no tenga un modelo 3D asociado este botón será sustituido por otro que permite iniciar el proceso de generación de nuevo.

En caso de que el usuario quiera editar la información relacionada con uno de los objetos, puede pulsar la opción de “editar” en la pantalla de detalles. Esto hará que se muestre una pantalla como la de la figura 47.

Desde esta nueva pantalla, el usuario puede modificar el nombre y la descripción del objeto. También es posible cambiar la imagen del objeto por una de la galería del dispositivo. Una vez realizados los cambios solo tiene que pulsar el botón de guardar para que la aplicación haga una petición al servidor y se registren todos los cambios.

Figura 47 Editar objeto

Adicionalmente, se han añadido transiciones entre las pantallas para ofrecer una experiencia más fluida al usuario y una pantalla de carga al inicio de la aplicación.

También se han incluido ventanas emergentes para informar al usuario sobre cualquier tipo de error que se pueda producir o solicitar los permisos necesarios, entre ellos, los permisos de la cámara.

Se ha desarrollado un sistema de notificaciones para informar al usuario de la finalización del proceso de generación de los objetos.

6.2.8 Arquitectura y despliegue

El proyecto sigue una arquitectura de cliente-servidor. El lado del cliente es la aplicación de móvil con la que el usuario puede interactuar. El lado del servidor, por otra parte, está formado por una *API* para la generación de objetos tridimensionales, un servicio *REST* y una base de datos. Cada uno de estos tres elementos se va a levantar dentro de un contenedor de *Docker* y todos ellos van a estar conectados a través de una red virtual de forma que puedan comunicarse.

El uso de *Docker* va a permitir lanzar los diferentes servicios independientemente del sistema operativo de la máquina sobre la que se ejecuten. Además, proporciona varias herramientas como *Docker Compose* para facilitar el lanzamiento de varios contenedores al mismo tiempo. En este caso, se han definido las reglas para la creación de cada contenedor y de la red virtual en un archivo *compose* que es utilizado a la hora del lanzamiento de la aplicación.

La aplicación de móvil se comunica tanto con el servicio *REST* como con la *API* de generación de objetos mediante el protocolo *http* a los puertos 9001 y 9000 respectivamente. La máquina donde se ejecuta el servidor mapea estos puertos directamente con los contenedores. Por último, el servicio *REST* está conectado a la base de datos y será el encargado de gestionar todas las peticiones de acceso o modificación de información. Todo esto se puede ver en el esquema de la figura 48.

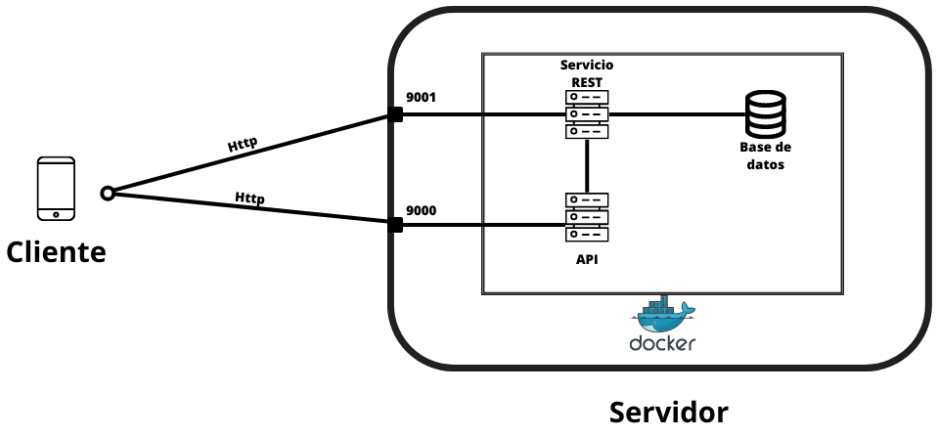


Figura 48 Esquema de despliegue

7. Resultados y discusión

Una vez ha finalizado el desarrollo del proyecto, se pueden analizar los resultados. Se han completado todos los objetivos principales establecidos al inicio junto con algunos de los objetivos adicionales.

Se ha desarrollado una aplicación de móvil que, mediante varias imágenes o un vídeo, tanto elegidos desde la galería como capturados en el momento a través de la aplicación, es posible generar un modelo tridimensional. El objeto generado se guarda junto con una serie de datos que pueden ser consultados, modificados o borrados. Los objetos generados se pueden visualizar en realidad aumentada mediante la aplicación, además, se pueden rotar, mover, cambiar de tamaño o eliminar de la escena en tiempo real.

En cuanto al lado del servidor, se ha desarrollado una *API* encargada de la generación de los modelos tridimensionales haciendo uso de inteligencia artificial y de la GPU de la máquina en la que se instale el proyecto. Por otro lado, el servicio *REST* y la base de datos proporcionan la capacidad de almacenar de forma persistente la información generada por el usuario permitiendo su fácil gestión.

A lo largo del desarrollo del proyecto se han adquirido una serie de habilidades nuevas que se consideran bastante relevantes, estas son:

- Programación con el lenguaje *Kotlin.*
- Desarrollo de aplicaciones móviles.
- Experiencia con tecnologías novedosas como la realidad aumentada.
- Aplicación de modelos de inteligencia artificial en proyectos reales.
- Mejora de habilidades de despliegue de aplicaciones y de contenedores *Docker*.
- Mayor fluidez en el uso de la tecnología *Git*.
- Manejo de espacios y objetos tridimensionales.

En cuanto líneas de trabajo futuras, hay un amplio margen de mejora. Estas son algunas de las posibles ampliaciones del proyecto:

- Desarrollo de una página web desde donde gestionar los objetos de cada usuario sin necesidad de la aplicación de móvil. Esta web también podría permitir iniciar la generación de objetos y visualizar los resultados finales.
- La posibilidad de visualizar el objeto tridimensional sin el uso de realidad aumentada a modo de previsualización y para ofrecer mayor compatibilidad en dispositivos en los que la realidad aumentada no esté disponible.
- Ampliación de la capacidad de generación de objetos. Se podría desarrollar colas de peticiones de generación y aumentar la capacidad computacional del lado del servidor con el uso de un mayor número de máquinas para reducir los tiempos de espera.
- Visualizar en tiempo real el proceso de entrenamiento. Esto podría ser interesante ya que permitiría al usuario más flexibilidad a la hora de gestionar la generación de los modelos tridimensionales.
- Editor de objetos 3D. La aplicación de móvil podría contar con un sencillo editor de objetos que permitiera recortar los objetos para eliminar los elementos no deseados.

8. Bibliografía

[1] «Fotogrametría - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Fotogrametr%C3%ADa (accedido 29 de abril de 2023).

[2] «Fotogrametría, escanear en 3D con una cámara fotográfica - Bitfab». https://bitfab.io/es/blog/fotogrametria/ (accedido 29 de abril de 2023).

[3] «Fotogrametría | Ministerio de Transportes, Movilidad y Agenda Urbana». https://www.mitma.gob.es/instituto-geografico-nacional/observacion-del-territorio/fotogrametria (accedido 29 de abril de 2023).

[4] «IKEA - Apps on Google Play». https://play.google.com/store/apps/details?id=com.ingka.ikea.app&hl=en&gl=US (accedido 1 de mayo de 2023).

[5] «Todo sobre la Realidad Aumentada | Innovae». https://www.innovae.com/la-realidad-aumentada/ (accedido 29 de abril de 2023).

[6] «Overview of ARCore and supported development environments | Google Developers». https://developers.google.com/ar/develop?hl=en (accedido 29 de abril de 2023).

[7] «ARCore with Kotlin. Kotlin ARcore from zero to …. more than… | by Jose Arteaga | Medium». https://medium.com/@jose01.arteaga/kotlin-arcore-49b7a234f7cf (accedido 29 de abril de 2023).

[8] «Volume ray casting - Wikipedia». https://en.wikipedia.org/wiki/Volume_ray_casting (accedido 29 de abril de 2023).

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, y R. Ng, «NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, pp. 405-421, mar. 2020, doi: 10.1007/978-3-030-58452-8_24.

[10] M. Tancik *et al.*, «Nerfstudio: A Modular Framework for Neural Radiance Field Development», feb. 2023, Accedido: 29 de abril de 2023. [En línea]. Disponible en: http://docs.nerf.studio/index.html

[11] «Spring Initializr». https://start.spring.io/ (accedido 29 de abril de 2023).

[12] «Scrum Tools for Agile Teams». https://www.jetbrains.com/youtrack/agile/scrum/ (accedido 29 de abril de 2023).

[13] «GitLab Documentation». https://docs.gitlab.com/ (accedido 29 de abril de 2023).

[14] «Política de ramificación en git – Ikzer Dev». https://ikzerdev.wordpress.com/2015/03/10/politica-de-ramificacion-en-git/ (accedido 29 de abril de 2023).

[15] «Android Basics in Kotlin course | Android Developers». https://developer.android.com/courses/android-basics-kotlin/course (accedido 29 de abril de 2023).

[16] «Retrofit». https://square.github.io/retrofit/ (accedido 29 de abril de 2023).

[17] «GitHub - square/moshi: A modern JSON library for Kotlin and Java.» https://github.com/square/moshi (accedido 29 de abril de 2023).

[18] «Coil». https://coil-kt.github.io/coil/#download (accedido 29 de abril de 2023).

[19] «GitHub - SceneView/sceneform-android: Sceneform Maintained is an ARCore Android SDK with Google Filament as 3D engine. This is the continuation of the

archived Sceneform». https://github.com/SceneView/sceneform-android (accedido 29 de abril de 2023).
[20] «GitHub - ux3d/2gltf2: To glTF 2.0 converter». https://github.com/ux3d/2gltf2 (accedido 29 de abril de 2023).
[21] «Descripción general de CameraX | Desarrolladores de Android | Android Developers». https://developer.android.com/training/camerax?hl=es-419 (accedido 3 de mayo de 2023).