TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

I3D, PROTOTIPO DE APLICACIÓN MÓVIL PARA CAPTURAR RECUERDOS DE VIAJES

Autor: Francisco Javier Blanco Jiménez

Grado: Ingeniería Informática

Director: D. Jorge Delgado García
Departamento del director: Ingeniería Cartográfica, Geodésica y Fotogrametría

Fecha: 20/06/2024

CREA

Universidad de Jaén

Departamento de Informática

Don Francisco de Asís Conde Rodríguez**.** Tutor del Trabajo Fin de Grado titulado: '**I3D, prototipo de aplicación móvil para capturar recuerdos de viajes**', que presenta Don Francisco Javier Blanco Jiménez, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Julio de 2024

El alumno:

BLANCO JIMENEZ FRANCISCO JAVIER - 77380162C

Firmado digitalmente por BLANCO JIMENEZ FRANCISCO JAVIER - 77380162C
Fecha: 2024.06.25 12:59:16 +02'00'

Francisco Javier Blanco Jiménez

El tutor:

CONDE RODRIGUEZ FRANCISCO DE ASIS - 24226298F

Firmado digitalmente por CONDE RODRIGUEZ FRANCISCO DE ASIS - 24226298F
Fecha: 2024.06.25 13:10:58 +02'00'

Francisco de Asís Conde Rodríguez

Agradecimientos

A mi familia, por toda vuestra ayuda en mi etapa universitaria.

A mi mentor, Juanma, por tus consejos y empeño en mí.

A mi tutor, Francisco, por tu paciencia y perseverancia.

A Juanma Jurado, por tu ayuda incondicional.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sin mención
Idioma	Español
Tipo	General
TFG en equipo	No
Autor/a	Francisco Javier Blanco Jiménez
Fecha de asignación	02/11/2023
Descripción corta	Viajar es una aventura que nos transporta más allá de nuestro entorno cotidiano, permitiéndonos explorar otras culturas y paisajes. Cada viaje es una oportunidad de ampliar horizontes y compartir experiencias y nos brinda recuerdos imborrables. Es muy frecuente que cada vez que el viajero descubre algo que le llama la atención, haga una fotografía para llevarse un recuerdo que le permita recrear ese momento de vuelta en casa. Sin embargo, en ocasiones las fotografías no capturan bien toda la esencia de eso que nos llamó la atención. ¿Y si fuese posible capturar imágenes mejoradas de las cosas que nos llaman la atención en los viajes, con la facilidad con la que les hacemos fotografías? Este trabajo fin de grado consiste en desarrollar un prototipo de aplicación móvil que permita capturar fácilmente representaciones mas ricas que las fotografías clásicas de elementos reales que podamos encontrar en nuestros viajes, para poder observarlos posteriormente apreciando todo su esplendor.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFG-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFG-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFG-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

1 ESPECIFICACIÓN DEL TRABAJO 15

1.1 INTRODUCCIÓN 15
1.2 OBJETIVOS DEL TRABAJO 16
1.3 ANTECEDENTES Y ESTADO DEL ARTE 16
1.4 DESCRIPCIÓN DE LA SITUACIÓN DE PARTIDA 18
1.4.1 Resumen de las deficiencias y carencias identificadas 18
1.5 REQUISITOS INICIALES 19
1.6 HIPÓTESIS Y RESTRICCIONES 20
1.7 ESTUDIO DE ALTERNATIVAS Y VIABILIDAD 20
1.7.1 Creando un algoritmo de fotogrametría 20
1.7.2 Librería Meshlab 21
1.7.3 Librería OpenSFM 21
1.7.4 Ejecución directa desde el hardware móvil 22
1.7.5 Arquitectura Cliente-Servidor con NerfStudio 22
1.8 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 24
1.9 MATERIAL Y MÉTODOS 24
1.9.1 Front-End 25
1.9.2 Back-End 25
1.9.3 Nerfstudio (paquetes necesarios) 26
1.10 TECNOLOGÍAS UTILIZADAS 27
1.10.1 Flutter/Dart 27
1.10.2 Android Studio 27
1.10.3 Python 27
1.10.4 NerfStudio 28
1.10.5 Servidor 28
1.10.6 Smartphone para pruebas 29
1.10.7 API para conectar el servidor y la app (FLASK) 29
1.11 METODOLOGÍA DE DESARROLLO DE SOFTWARE 29
1.12 ANÁLISIS DE RIESGOS 30
1.13 ESTIMACIÓN DEL TAMAÑO Y ESFUERZO 31
1.14 PLANIFICACIÓN TEMPORAL 32
1.14.1 Fase inicial (25% del total) 32
1.14.2 Fase de diseño (30% del total) 32
1.14.3 Fase de implementación (45% del total) 32
1.15 PRESUPUESTO 33
1.15.1 Coste hardware 33

1.15.2 Coste humano 34
1.15.3 Mecanismos de control de calidad 36

2 DISEÑO INICIAL 36

2.1 ESPECIFICACIONES DEL SISTEMA 36
2.1.1 Requisitos funcionales 38
2.1.2 Requisitos no funcionales 38
2.1.3 Casos de uso 39
2.2 ANÁLISIS Y DISEÑO DEL SISTEMA 46
2.2.1 Diseño de la infraestructura del sistema 46
2.2.2 Diseño de las bases de datos 47
2.2.3 Diagrama de clases 51
2.2.4 Diagrama de casos de uso 52
2.2.5 Diagramas de secuencia 53
2.2.6 Diseño de la interfaz y storyboards 56

3 DESARROLLO 59

3.1 1529-01-IT1 60
3.2 2912-02-IT2 61
3.3 1224-02-IT3 62
3.4 2409-03-IT4 64
3.5 0923-03-IT5 65
3.6 2306-04-IT6 66
3.7 0620-04-IT7 68
3.8 2004-05-IT8 68
3.9 0418-05-IT9 68
3.10 1801-06-IT10 69
3.11 0115-06-IT11 70
3.12 PRUEBAS FINALES 70
3.12.1 Pruebas de verificación del sistema 70
3.12.2 Validación de la usabilidad del sistema 70

4 MODELO DE NEGOCIO 71

4.1 OBJETIVOS 71
4.1.1 Objetivos cualitativos 71
4.1.2 Objetivos cuantitativos 71
4.2 ANÁLISIS DEL ENTORNO 72
4.3 PLAN DE MERCADOTECNIA 73

5 CONCLUSIONES Y TRABAJOS FUTUROS 77

6 APÉNDICES 78

6.1 GUÍA ORIGINAL DEL TRABAJO FIN DE TÍTULO 78

6.1.1 Descripción corta 78

6.1.2 Objetivos 79

6.1.3 Metodología a desarrollar 79

6.1.4 Documentos y formato de entrega 81

6.2 INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA 81

6.2.1 Puesta en marcha del servidor 81

6.2.2 Instalación de la aplicación 84

6.3 MANUALES DE USUARIO 85

6.3.1 Inicio de sesión 85

6.3.2 Crear cuenta 86

6.3.3 Verificar cuenta 87

6.3.4 Crear modelo con cámara 88

6.3.5 Visualizar modelo local 90

6.3.6 Subir modelo 91

6.3.7 Eliminar modelo local 92

6.3.8 Descargar modelo 93

7 DEFINICIONES Y ABREVIATURAS 94

8 BIBLIOGRAFÍA 95

Índice de ilustraciones

Ilustración 1 - Polycam 17
Ilustración 2 - Coincidencias del descriptor SIFT 21
Ilustración 3 - Nerfstudio 23
Ilustración 4 - Metodología de desarrollo ágil 30
Ilustración 5 - Diseño de la infraestructura 47
Ilustración 6 - Firebase Authentication 48
Ilustración 7 - Firebase Firestore 49
Ilustración 8 - Firebase 49
Ilustración 9 - Diagrama de Firebase 50
Ilustración 10 - Diagrama SQL 51
Ilustración 11 - Diagrama de clases 51
Ilustración 12 - Diagrama de secuencia F1 53
Ilustración 13 - Diagrama de secuencia F2 y F3 54
Ilustración 14 - Diagrama de secuencia F4, F5, F6, F7 y F8 55
Ilustración 15 - Diagrama de secuencia F11, F12 y F13 56
Ilustración 16 - Storyboard Inicio y Login 57
Ilustración 17 - Storyboard Pantalla Principal 57
Ilustración 18 - Storyboard Pantalla Mis Modelos 58
Ilustración 19 - Storyboard Cloud 58
Ilustración 20 - Logo I3D 59
Ilustración 21 - Carpeta raíz del servidor 61
Ilustración 22 - Error nerfstudio GitHub 62
Ilustración 23 - Solución error versiones 62
Ilustración 24 - Nube de puntos 63
Ilustración 25 - Archivo firebase_options 64
Ilustración 26 - Archivo Manifest 65
Ilustración 27 - Archivo DatabaseHelper 66
Ilustración 28 - Plantilla correo de verificación 67
Ilustración 29 - Carpeta Servidor con correo 67
Ilustración 30 - Análisis DAFO 72
Ilustración 31 - Interés a lo largo del tiempo de palabras clave 77
Ilustración 32 - Instalación CUDA 82
Ilustración 33 - Web oficial Nerfstudio 83
Ilustración 34 - Activar orígenes desconocidos 84
Ilustración 35 - Manual de usuario - Inicio de sesión 85
Ilustración 36 - Manual de usuario - Crear cuenta botón 86
Ilustración 37 - Manual de usuario - Crear cuenta 87
Ilustración 38 - Manual de usuario - Verificar email 87
Ilustración 39 - Manual de usuario - Crear modelo cámara 88
Ilustración 40 - Manual de usuario - Botón aceptar crear modelo 89
Ilustración 41 - Manual de usuario - Botón comenzar construcción 89
Ilustración 42 - Manual de usuario - Icono carpeta 90
Ilustración 43 - Manual de usuario - Seleccionar modelo 90

Ilustración 44 - Manual de usuario – Visualizado.....91
Ilustración 45 - Manual de usuario - Subir modelo.....91
Ilustración 46 - Manual de usuario - Eliminar modelo 92
Ilustración 47 – Manual de usuario - Descargar modelo.....93
Ilustración 48 - Manual de usuario - Ver modelo descargado.....93

Índice de tablas

Tabla 1 - Costes componentes servidor 33
Tabla 2 - Costes componentes ordenador 34
Tabla 3 - Coste humano 36
Tabla 4 - Requisitos funcionales 38
Tabla 5 - Requisitos no funcionales 39
Tabla 6 - Caso de uso F1 39
Tabla 7 - Caso de uso F2 40
Tabla 8 - Caso de uso F3 40
Tabla 9 - Caso de uso F4 41
Tabla 10 - Caso de uso F5 41
Tabla 11 - Caso de uso F6 42
Tabla 12 - Caso de uso F7 42
Tabla 13 - Caso de uso F8 43
Tabla 14 - Caso de uso F9 43
Tabla 15 - Caso de uso F10 44
Tabla 16 - Caso de uso F11 44
Tabla 17 - Caso de uso F12 45
Tabla 18 - Caso de uso F13 45
Tabla 19 - Colores personalizados 69
Tabla 20 - Objetivos cualitativos 71
Tabla 21 - Objetivos cuantitativos 71
Tabla 22 - Tipos de monetización 73
Tabla 23 - Viabilidad de monetización 74

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 7 (Definiciones y abreviaturas).

1.1 Introducción

Actualmente, todo teléfono móvil lleva integrada una cámara capaz de capturar imágenes con resoluciones que, en muchos casos, mejoran las de cámaras de alta potencia, pudiendo guardar en la galería recuerdos o capturas de instantáneas para uso profesional. Pero, podremos llevar esto un paso más adelante y, con la misma filosofía, realizar esta lógica con modelos en tres dimensiones, un acercamiento más al futuro en la palma de nuestra mano.

El objetivo principal de este Trabajo de Fin de Grado es hacer de tu teléfono un entorno capaz de procesar imágenes y crear modelos en tres dimensiones de cualquier objeto que se desee, capturando unas cuantas imágenes alrededor de él.

La elaboración del proyecto tiene como razón principal el facilitar al usuario una aplicación que, no sólo satisfaga sus necesidades (ya cubiertas dentro del ámbito de las fotos instantáneas), sino, de crear una necesidad para el usuario final mostrando la eficiencia y satisfacción que puede llegar a brindarle la fácil creación de modelos en 3D en pocos pasos.

A lo largo del documento se observará detalladamente las distintas alternativas, así como los pasos seguidos para conseguir un correcto funcionamiento del sistema, junto con las tecnologías utilizadas, algoritmos implicados, funciones matemáticas necesarias o el propio proceso de creación. Es un proyecto que cumple perféctamente con las primitivas para convertirse en un competidor futuro de aplicaciones similares.

1.2 Objetivos del trabajo

- Reforzar los conocimientos adquiridos durante el grado sobre desarrollo de aplicaciones multiplataforma para dispositivos móviles.
- Aprender como realizar modelos en tres dimensiones de objetos, y cómo representarlos mediante técnicas de realidad aumentada
- Estudiar las condiciones más habituales de las personas que viajan: entorno de uso de móvil, tiempo disponible para crear un recuerdo tridimensional, condiciones ambientales, capacidad de recargar dispositivos, etc.
- Estudiar las necesidades más importantes de las personas que viajan a la hora de capturar recuerdos de los elementos de interés que encuentran durante el viaje.
- Elegir la mejor arquitectura para desarrollar el prototipo de la aplicación móvil
- Desarrollar un prototipo de aplicación móvil para capturar recuerdos tridimensionales, teniendo en cuenta la experiencia de uso

1.3 Antecedentes y estado del arte

Actualmente, es muy común el uso de modelos en tres dimensiones para distintos fines, ya sea para reconstrucción de obras de arte, diseño de modelos para películas o videojuegos, reconstrucción de escenas del crimen o para diseño arquitectónico. El propio concepto de “Tecnología 3D” tiene su inicio en la década de los 50 y los 60, con la creación de modelos tridimensionales muy simples sin ningún tipo de textura. Posteriormente, en la década de los 80 se empezaron a utilizar las primeras aplicaciones de diseño 3D (muy lejos de las que usamos hoy día).

En los años 90 se comenzó a aplicar este método para la ingeniería, arquitectura y otros campos como la medicina (Softimage). Finalmente, en la década de los 2000 conseguimos un avance a gran escala aportando un realismo a los modelos 3D superior, llegando, en los últimos años a la impresión 3D y la revolución que consigo lleva.

Hasta hace pocos años, no existía ninguna aplicación móvil de propósito general capaz de crear y renderizar modelos 3D, hasta que, el 3 de abril de 2022, se lanzó una aplicación al mercado llamada “Polycam”, ofrecida por Google Commerce

Ltd. y capaz de crear modelos tridimensionales a partir de un simple video (el objetivo que perseguimos). Esta aplicación cuenta con servidores muy potentes capaces de generar los modelos con una rapidez abrumadora. Indagando más en la aplicación, proporciona una red social donde los usuarios pueden ver y subir sus modelos generados inApp.

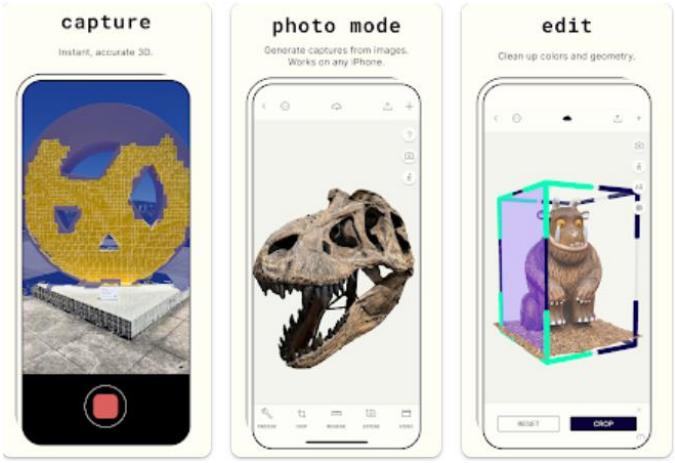


Ilustración 1 - Polycam

De la misma forma, aunque no se traten de aplicaciones para dispositivos móviles, existen aplicaciones web capaces de crear modelos pasando unas pocas imágenes. Un claro ejemplo de aplicación web es www.3d.csm.ai

Existen otras aplicaciones móviles, tales como Trnio, Qlone, Scann3D o Cappasity, con las mismas capacidades, pero menos famosas. Un punto importante a tener en cuenta es que la mayoría de ellas solo están disponibles para IOs, y tiene una clara explicación, el sensor LIDAR de las cámaras de Apple. Este sensor, cuyas siglas describen Light Detection and Ranging, utiliza una tecnología que detecta y mide mediante una luz láser distancias y crea mapas tridimensionales del entorno.

Por lo tanto, con la aparición de este sensor en las cámaras Apple se abrió un nuevo mercado de aplicaciones que está todavía en auge y que nuevas marcas están añadiendo a sus SmartPhones, como Google (Pixel) o Samsung. El sensor emite un pulso, lo refleja y detecta para realizar el cálculo de la distancia del pulso reflejado mediante la ecuación presentada y por último generar el mapa 3D:

$$\text{Distancia} = \frac{\text{Velocidad de la luz x Tiempo de retorno}}{2}$$

Para comprender cómo estas aplicaciones , a partir de unas cuantas imágenes o un video, son capaces de crear los modelos, debemos comprender de la misma forma la fotogrametría que hay detrás de las mismas y se verá en apartados posteriores.

1.4 Descripción de la situación de partida

Como se ha descrito en la introducción, se mejorará un aspecto importante en los sistemas de la información actuales, más concretamente el campo de la fotografía, intentando cambiar la visión de lo que conocemos por fotografía e intentar dar un giro a este mismo. Este software aportará un nuevo cambio.

Al igual que las demás aplicaciones existentes, será necesario el uso de un servidor para la ejecución del algoritmo de modelado 3D, una base de datos para almacenamiento de usuarios, una base de datos en local para almacenamiento de modelos en 3D, diferentes librerías de gestión de visualización de modelos tridimensionales así como librerías para gestión de cámara y archivos, API para conectar el servidor con el cliente e interfaz de usuario.

1.4.1 Resumen de las deficiencias y carencias identificadas

En la mayoría de las aplicaciones analizadas se encontró un error común en varias de ellas, la falta de una gestión de usuarios. Cuando hablamos de gestión de usuarios nos referimos a un sistema de inicio de sesión para dejar constancia de quien está utilizando la aplicación. Esto es de vital importancia por varias razones:

- Seguridad
- Privacidad
- Control de versiones
- Monitorización de la aplicación
- Recolección de datos

Todas estas razones conforman partes muy importantes en un software, tanto para el cliente como para la empresa dueña del mismo. Si no tenemos un control de quien está usando nuestra aplicación, no podremos determinar cómo mejorarlo, qué

errores encontramos, si están entrando falsos usuarios o bots en la aplicación y muchos más errores que pueden perjudicar el futuro del software.

Otro error común es el soporte para generación de modelos a partir de vuelos de drones, muchas de las aplicaciones, no lo tienen. Hoy en día es un sector claramente en auge debido a su capacidad de resolución de problemas y de eficiencia en el tiempo, por lo que, si una aplicación de modelado 3D permite esta sincronización con vuelos de drones para un posterior modelado, se estaría contribuyendo a mejorar una parte crucial del sector tecnológico de la modelación tridimensional.

Por otra parte, también se encuentra el mayor problema presentado y con solución complicada, los altos tiempos de carga dentro de las aplicaciones. Cuando un software ejecuta un algoritmo de Deep Learning, necesita de una infraestructura hardware de mucho nivel y potencia para evitar tiempos de carga excesivos. En aplicaciones como Polycam, estos tiempos son muy reducidos ya que la capacidad de sus servidores es mucho mayor que en otras. El mayor inconveniente de esto es el capital necesario para que estos tiempos de carga sean menores, cuando una aplicación ejecuta un algoritmo de tecnología 3D en un servidor, por norma general esto necesita de una capacidad gráfica muy elevada, y, como ya se conoce, el mercado de tarjetas gráficas es muy caro.

Para terminar esta sección, repasemos las carencias identificadas:

- Gestión de usuarios
- Sincronización con vuelo de drones
- Tiempos de carga excesivos

1.5 Requisitos iniciales

Como tratamos un tema software dirigido al público general, definir unos requisitos iniciales que éste debería de cumplir es una tarea importante y necesaria para conseguir un correcto funcionamiento y por ende un posible éxito de la aplicación.

- Permitir la generación de modelos tridimensionales
- Gestión de usuarios
- Gestión de modelos en bases de datos
- Comunicación con el servidor

1.6 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

1.7 Estudio de alternativas y viabilidad

1.7.1 Creando un algoritmo de fotogrametría

Inicialmente se trató de realizar un algoritmo capaz de generar nubes de puntos para posteriormente tratarlas usando código Python plano. Llegando al punto de conseguir detectar coincidencias entre pares de imágenes a partir un detector (SIFT) inicialmente con sus puntos de referencia y los correspondientes descriptores para cada imagen. Continuando con un BFMatcher para detectar coincidencias de puntos en los pares de imágenes (alrededor de 28 imágenes de prueba).

Una vez detectadas las coincidencias de los pares de imágenes, se dibujaban sobre otras nuevas imágenes para poder ver el resultado y analizarlo, observando que había muchos puntos coincidentes erróneos por su similitud. También se intentó adivinar la matriz de homografía y la matriz fundamental de la cámara (necesario para su posterior calibración), pero, al no tener un sensor (LIDAR) en la cámara usada, era un proceso tedioso y largo. Teóricamente, al haber realizado los pasos descritos, podríamos calibrar la cámara y generar la nube de puntos, pero no se llegó a determinar un espacio lo suficientemente exacto para la construcción de esta nube de puntos.

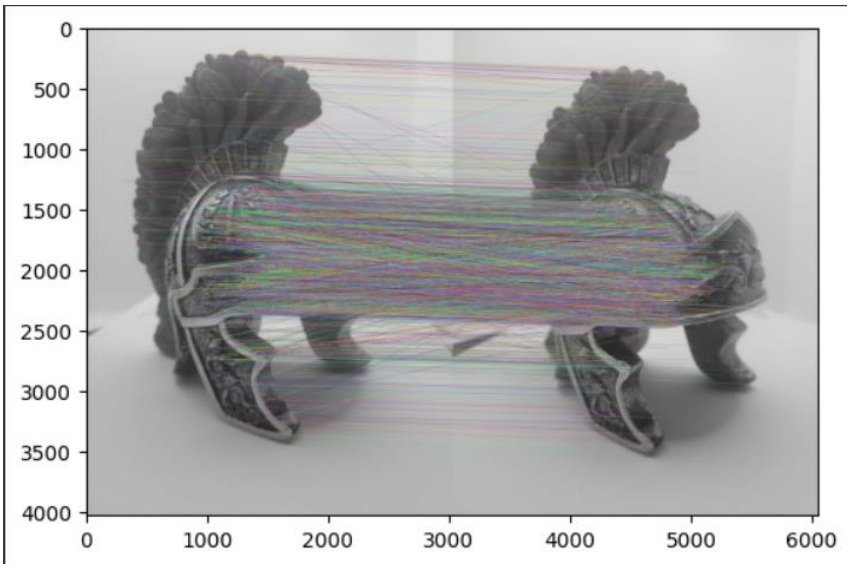


Ilustración 2 - Coincidencias del descriptor SIFT

En la imagen anterior se puede observar las coincidencias descritas, teniendo hasta más de 10000 coincidencias, de las que muchas de ellas fallaban.

1.7.2 Librería Meshlab

Meshlab es un software libre utilizado para el tratamiento de modelos en tres dimensiones, disponible en GitHub con código abierto. Tratando de buscar nuevas formas para generar modelos, se navegó por GitHub llegando a (AliceVision) que proporcionaba una API así como un software capaz de tratar modelos en 3D y de generarlos a partir de imágenes. Con ello se halló un problema en la API ya que era incapaz de ejecutarse por razones que se desconocen (posiblemente por el Sistema Operativo anfitrión usado). Pero, haciendo pruebas se generaba el modelo con total exactitud usando el software que ofrecen (solo disponible para ordenador).

1.7.3 Librería OpenSFM

OpenSFM estuvo a punto de ser la librería que determinaría el Trabajo de Fin de Grado, con un inconveniente que marcó su declive dentro de este trabajo, la incompatibilidad que tiene con Windows 11 (el Sistema Operativo en el ordenador que se ejecuta). Como en este punto no se tuvo en cuenta un posterior servidor que realizase los modelos con un sistema Linux, se dejó de lado.

OpenSFM ofrece una amplia variedad de opciones para generación de modelos en tres dimensiones, modelos geométricos, algoritmos de reconstrucción incrementales… todo ello desde línea de comandos y/o desde código Python.

1.7.4 Ejecución directa desde el hardware móvil

Una opción probada fue la ejecución del código dentro de la propia aplicación móvil localmente en el teléfono que la ejecutase, encontrando los siguientes problemas:

- Librería de Python muy simple para su uso en Flutter con restricciones muy marcadas
- Dart no está lo suficientemente desarrollado ni a tan alto nivel para aceptar librerías de fotogrametría
- Ralentización de la aplicación
- Tiempos de carga excesivos
- Errores de ejecución y caídas

Estos errores en gran medida se deben a la incapacidad de los teléfonos actuales de ejecutar algoritmos demasiado elaborados debido a la gran diferencia de hardware presentada entre su fin inicial (ejecución en ordenadores potentes) y su final (ejecución en móviles, en los que cabe la posibilidad de que el usuario no tenga un teléfono potente y, de la misma forma, caro) ya que se intenta crear la aplicación para todo el público posible.

1.7.5 Arquitectura Cliente-Servidor con NerfStudio

La opción adoptada, tras haber probado todos las opciones anteriormente expuestas, y tras intentar responder a la pregunta ¿Cómo se consigue ejecutar un programa con altos requerimientos hardware en un dispositivo móvil?, se decidió utilizar un servidor con unas características capaces de ejecutar mediante gráfica el framework que finalmente sería capaz de generar modelos en 3D.

Inicialmente la Universidad de Jaén proporcionó un servidor para intentar hacer esta arquitectura desde una máquina con las siguientes especificaciones:

- 8 GB de RAM

- Tarjeta gráfica [AMD/ATI] ES1000
- Disco duro sólido de 120 GB
- Puertos HTTP y SSH abiertos

A primera vista, puede ser un sistema con características de una calidad aceptable, pero, insuficientes para el marco que este TFG ocupa. Por lo tanto, tras varias pruebas, habiendo configurado íntegramente el servidor, con NGINX como cliente para recibir las solicitudes (proxy), y FLASK como aplicación que se ejecutaría desde el lado del servidor, no se consiguió un correcto funcionamiento del mismo, y como principal razon encontramos la tarjeta gráfica, que, al pertenecer a AMD, no es compatible con el sistema que el framework usado (Nerfstudio) utiliza, llamado CUDA (perteneciente a NVIDIA).

Ilustración 3 - Nerfstudio

Habiendo pasado esta experiencia, y con el sistema funcionando en una máquina personal, se tomó la decisión de convertir dicha máquina en el servidor que estaría detrás de todo el proceso de reconstrucción 3D. Este PC se trata de un ordenador portátil MSI GF63 thin, que, mientras se realiza el prototipo cumplirá con las expectativas con muy buenos resultados, pudiendo en un futuro trasladar la arquitectura a un servidor más potente.

El servidor será el encargado de, recibir las solicitudes POST de la aplicación y procesar las imágenes para finalmente generar y devolver el modelo en tres dimensiones a la aplicación. Para ello se ha hecho uso del framework ya mencionado

Nerfstudio. Nerfstudio es una librería compatible con código Python capaz de, entre otras cosas, generar nubes de puntos, modelos en 3D…. Se basa en un sistema de red neuronal con geometría epipolar que recibe una entrada (imágenes, vídeo…), para entrenar el modelo y generar una salida (video 3D, modelo 3D, nube de puntos, mesh…).

1.8 Descripción de la solución propuesta

Habiendo estudiado con ahínco todas las diferentes alternativas posibles, se llegó a la conclusión de crear una infraestructura con front y back de diferentes maneras y con las siguiente tecnologías:

- Flutter para realizar la aplicación que se encuentra en el front y proporcionar una aplicación fácil de usar y multiplataforma
- Servidor para la ejecución del algoritmo de Nerfstudio de generación de modelos tridimensionales
- API con Flask para proporcionar una conexión entre el servidor y la aplicación
- Base de datos con Firebase para la gestión de usuarios en la nube, proporcionando una alta monitorización de las tareas que se realizan y una seguridad alta al pertenecer este servicio a Google
- Base de datos local para la gestión de modelos en el dispositivo que ejecuta la aplicación

Todo esto unido y trabajando de una forma sincronizada permitirá que se consiga el cambio importante que se mencionó en la introducción, y dar un paso más para acercarnos a la nueva generación de “capturas instantáneas”.

1.9 Material y métodos

Como material utilizado, se explicará cada una de las librerías involucradas en todo el TFG junto con una breve explicación de su uso, tanto para la parte de front como para el back.

1.9.1 Front-End

Las siguientes librerías proporcionadas pertenecen a Flutter, y todas ellas se encuentran disponibles para su uso y consultas en www.pub.dev

- http: necesario para la conexión TCP con el servidor
- camera: permite el uso de la cámara del dispositivo para la toma de imágenes o video
- provider: importante para la gestión inApp usando el conocido provider – consumer
- file_picker: proporciona la capacidad de seleccionar archivos de la galería
- fan_carousel_image_slider: necesario para el mini tutorial de cómo realizar el modelo 3D
- firebase_core: librería de firebase para la conexión con la base de datos en la nube
- path_provider: detección y uso de paths
- path: librería adicional a path_provider
- flutter_3d_controller: permite la visualización con OpenGL de modelos 3D
- sqflite: gestión de base de datos local con SQL
- firebase_auth: dentro de la librería de Firebase, proporciona la autentificación de usuarios
- image_picker: permite seleccionar imágenes y video del dispositivo

1.9.2 Back-End

Las librería proporcionadas a continuación se encuentran en el servidor, y todas ellas conforman el back-end y posibilitan la ejecución de la aplicación Flask para el modelado 3D.

- Nerfstudio: la más importante, permite la creación de modelos 3D con un algoritmo basado en redes neuronales y aprendizaje profundo

- Flask: encargada de poner en marcha el servidor en las interfaces requeridas, internamente permite enviar archivos, recibir solicitudes POST y GET o leer archivos json
- Os: permite la ejecución de comandos así como la gestión de los archivos en el servidor
- Subprocess: ejecución de subprocesos en terminales desde la propia aplicación flask
- Trimesh: conversión de archivos .obj a .glb
- Zipfile: compresión de archivos para su posterior envío a la aplicación

1.9.3 Nerfstudio (paquetes necesarios)

Para hacer posible que Nerfstudio ejecute sin ningún problema es necesario instalar los siguientes paquetes:

- Python: lenguaje que utiliza Nerfstudio
- Git: para clonar el repositorio de Nerfstudio
- CUDA: paquete que utiliza NVIDIA para realizar tareas de computación general, por lo que es necesario disponer de una tarjeta gráfica de NVIDIA
- cuDNN: paquete adicional de CUDA
- Microsoft Visual C++ Build Tools
- (opcional) Conda: se puede crear un entorno con anaconda para encapsular todos estos paquetes
- Torch; perteneciente a PyTorch, proporciona una biblioteca de aprendizaje automático
- Torchvision: perteneciente también a PyTorch
- Tensorboard: herramienta de visualización de datos para el flujo de trabajo del aprendizaje automático
- OpenCV: librería utilizada también por Nerfstudio para el aprendizaje automático

1.10 Tecnologías utilizadas

Para el desarrollo de este TFG se han utilizado las siguientes tecnologías:

- Flutter/Dart
- Android Studio
- Python
- NerfStudio (usa algoritmos de fotogrametría y geometría epipolar)
- Servidor
- Teléfono móvil para las pruebas de la aplicación (Samsung Galaxy S20FE)
- API para conectar el servidor con la aplicación

1.10.1 Flutter/Dart

Flutter junto con Dart constituyen un lenguaje de programación orientado al desarrollo de aplicaciones móviles usando Widgets para la interfaz gráfica y compatibles con la programación orientada a objetos. Flutter es una tecnología creada por Google que la hace sincronizable con todos los servicios de Google, así como, Maps, Drive, Firebase…

Lo más importante de flutter es el desarrollo multiplataforma y su capacidad de adaptación del código tanto para IOs, Android, web, relojes inteligentes o SmartTVs. Al crear el proyecto con Flutter tienes la posibilidad de elegir las plataformas en las que estará disponible la aplicación, seleccionándolo desde un menú para así proporcionar los archivos necesarios (como la carpeta para IOs).

1.10.2 Android Studio

Es un IDE enfocado a el desarrollo de aplicaciones móviles, la base en la que se sustenta Flutter en este trabajo (también existe la posibilidad de usar Flutter en Visual Studio). A la hora de exportar una aplicación, simplemente con pulsar en “Build Release” ya se genera el archivo .apk de la aplicación con todas las librerías necesarias que se hayan instalado dentro del proyecto.

1.10.3 Python

Se ha elegido Python por ser un lenguaje de alto nivel y por su variedad de librerías disponibles que hacen posible el desarrollo de este TFG, junto con su clara

compatibilidad con el sistema Windows (ya que Python será el lenguaje que se use en el servidor cuando se ejecute el script necesario para generar el modelo en tres dimensiones).

1.10.4 NerfStudio

Principal librería que ayuda en el proceso de generación del modelo, con altas capacidades, incluso tiene la posibilidad de generar modelos a partir de vuelos de drones. En el caso de este proyecto, se puede generar un vídeo de recorrido del objeto (las fotos que se han ido realizando alrededor del mismo), y de igual forma, generar la nube de puntos pudiendo exportarla para tratarla posteriormente.

Se basa en una red neuronal previamente entrenada con modelos propios o los brindados por la comunidad. Sus métodos principales (incluidos en la documentación de su web) son:

- Nerfacto: Método que integra múltiples métodos en uno
- Instant-NGP: Primitivas de gráficos neuronales con codificación hash multiresolución
- NeRF: Campos de radiación neuronal OG
- Mip-NeRF: Representación multiescala para campos de radiación neuronal antialiasing
- TensoRF: Campos de radiación tensorial
- Splatfacto: Implementación de Gaussian Splatting de NerfStudio

Tiene a su vez compatibilidad con Blender, lo que puede dar mucho juego a la hora de exportar modelos.

1.10.5 Servidor

Este servidor actuará como Back-end para ejecutar el algoritmo que no puede un smartphone normal hoy en día (por incapacidad hardware). Las especificaciones hardware del servidor son:

- 16 Gb de Memoria principal
- Procesador Intel Core i7 de 10ª generación
- Sistema operativo Windows 11

- Disco duro de 500 GB (necesarias para alojar modelos de entrenamiento de la red neuronal)
- Puertos HTTP abiertos para poder hacer la arquitectura client to side

1.10.6 Smartphone para pruebas

Es necesario un smartphone para realizar las pruebas físicas (aunque Android Studio tenga sus propios emuladores) y ejecutar la aplicación en busca de errores o bugs que puedan afectar a la aplicación, así como un buen posicionado de los elementos en la pantalla (usando diferentes resoluciones en los emuladores y el teléfono físico para comprobar si es responsive).

1.10.7 API para conectar el servidor y la app (FLASK)

Para conectar la aplicación del front con el servidor del back es necesario programar una API que permita esta conexión. Se usará Flask con Python para poder permitir este enlace, constará de distintos métodos POST a los que la aplicación podrá llamar para ejecutar distinto código, como:

- Guardar imágenes o video en el servidor
- Procesar imágenes
- Entrenar el modelo
- Crear el archivo .obj
- Transformar el .obj a .glb (la extensión permitida en el front)
- Enviar el modelo de vuelta a la aplicación

1.11 Metodología de desarrollo de software

Para realizar este TFG se ha utilizado una metodología de desarrollo ágil, permitiendo un proceso de entrega por partes, donde el cliente puede ver a cada iteración un resultado y así poder abordar problemas de una manera más eficiente dejando que el cliente de una visión sobre lo hasta cierto momento del tiempo realizado y no perder tiempo desarrollando algo que es probable que en un futuro el cliente quiera que se realice de una forma distinta.

Al ser un TFG, es un trabajo individual, por lo que se deberá abordar esta metodología de desarrollo desde un punto de vista distinto al tradicional (con un equipo de desarrollo).

Las distintas fases que componen un desarrollo ágil son las siguientes, comenzando con un plan:

- Diseño
- Desarrollo
- Test
- Despliegue
- Repaso

Ilustración 4 - Metodología de desarrollo ágil

Todas estas fases se van completando a cada iteración (método circular), para en una iteración final, lanzar la aplicación y entregarla al cliente.

1.12 Análisis de riesgos

Como se ha mostrado en el apartado de estudio de alternativas y viabilidad, se han encontrado distintos riesgos que ha sido necesario mitigarlos y buscar alternativas, dichos riesgos se enumeran a continuación:

- El algoritmo de modelado al programarlo necesitaba de una calibración previa de la cámara, para calibrar una cámara es necesario (en su forma tradicional) de un tablero de ajedrez, y, como es lógico, no se va a obligar

al usuario a poseer un tablero para poder calibrar la cámara y permitir así generar el modelo, por lo tanto, se pasó a OpenSFM

- OpenSFM es incompatible con Windows, se debió buscar una alternativa a esta librería, que llegaría a ser MeshLab
- Meshlab, una librería que necesita CUDA (al igual que Nerfstudio), generaba una cantidad de problemas considerables como para encontrar la necesidad de utilizar otro paquete, este paquete llegaría a ser Nerfstudio
- Aunque Nerfstudio acabará siendo el paquete utilizado en este TFG, es de vital importancia destacar el cuidado que hay que tener a la hora de instalarlo, ya que, todos los paquetes necesarios, descritos en la sección Material y métodos necesitan tener instalada una versión concreta para que no haya incompatibilidades de versiones. Por ejemplo, tiny cuda necesita de una versión concreta de PyTorch, la 1.8, para poder ejecutar en armonía con el mismo
- Otro problema importante es la infraestructura utilizada. Al inicio del TFG se valoró utilizar un servidor que, al no cumplir con los requisitos mínimos, fue imposible utilizarlo para el modelado, y se llegó a la conclusión de usar otro servidor con capacidades mayores
- Por último, este servidor nuevo, tenía unas altas prestaciones pero, contando con una RAM de 16 Gb que al ejecutar el algoritmo de reconstrucción, ocuparía toda la memoria disponible, obligando a aumentar la memoria principal del sistema a 32 Gb

1.13 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

1.14 Planificación temporal

A continuación se presenta el diagrama de Gantt de la planificación de este TFG, comenzado en Enero de 2024. Al usar una metodología de desarrollo ágil, no hay que tener una planificación exacta, pero los puntos más importantes son los siguientes, indicando el porcentaje de tiempo invertido:

1.14.1 Fase inicial (25% del total)

- Análisis de librerías de reconstrucción
- Estudio de frameworks
- Estudio de IDE
- Análisis de requisitos
- Pruebas con algoritmos
- Pruebas con diferentes APIs
- Pruebas con servidor
- Estimación de coste

1.14.2 Fase de diseño (30% del total)

- Diseño de la aplicación con Flutter
- Diseño del Back
- Diseño de la API de comunicación

1.14.3 Fase de implementación (45% del total)

- Implementación de la app con Flutter
- Implementación del servidor Flask
- Implementación de la API
- Pruebas con modelos

1.15 Presupuesto

Al tratarse de un producto software con necesidades hardware altas y una clara necesidad de intervención de varios expertos en la tecnología a desarrollar, podremos desglosar los costes en humanos y hardware, teniendo en cuenta que se usan herramientas de implementación gratuitas como pueden ser Flutter, Nerfstudio o Android Studio.

1.15.1 Coste hardware

Para tener un sistema consistente y con una eficiencia notable, será necesario invertir buena parte del presupuesto en obtener una infraestructura adecuada tanto para los desarrolladores del sistema como para que los clientes obtengan un tiempo de respuesta de la aplicación de calidad, con ello, será necesario un servidor con los siguientes componentes:

Componente	Precio
Tarjeta gráfica NVIDIA 1650 Ti GTX	162,99 €
Procesador Intel Core i7 10ª Gen.	264,02 €
Memoria principal de 32 Gb	66,99 €
Disco duro sólido (SSD) de 500 Gb	46,99 €
Tarjeta ethernet	33,99 €

Tabla 1 - Costes componentes servidor

Así, sumando todos los componentes, llegaríamos a un total de 575,95 € de coste del servidor.

De la misma manera, se debe estimar el coste total del equipo utilizado para el desarrollo e implementación de la aplicación, teniendo en cuenta que el programador necesita un ordenador con altas preferencias para poder trabajar de la forma más eficiente posible:

Componente	Precio
Procesador Intel core i7 12ª Gen.	439,99 €
Tarjeta gráfica NVIDIA RTX 3060	320,99 €

Memoria principal de 32 Gb	66,99 €
Disco duro sólido (SSD) de 1 Tb	88,99 €
Tarjeta ethernet	33,99 €
Teclado y ratón	100 €
2 Monitores MSI 75 Hz	538,99 €
Componentes adicionales (caja, auriculares…)	200 €

Tabla 2 - Costes componentes ordenador

En total, tendríamos un coste en infraestructura de programación de 1789,94 € para permitir un desarrollo sin problemas ni retardos, suponiendo un uso total del 10% de la vida útil del equipo, serían 178.99€.

Sumando el coste del servidor al actual coste del ordenador empleado, tendríamos un total de 754.94 € en infraestructura.

1.15.2 Coste humano

Una aplicación de este calibre necesita de la intervención de distintos profesionales para hacer posible su correcto funcionamiento. Para ello, lo dividiremos en las distintas fases que tiene un proyecto software (hay que tener en cuenta que esto es una estimación ya que el proyecto se ha realizado por una única persona.

1.15.2.1 Fase de Preparación

- Gerente de producto, para definir los requisitos, gestionar el proyecto y coordinar los equipos
- Analista de negocios, analiza las necesidades de negocio y convertirlas en requisitos iniciales

1.15.2.2 Fase de diseño

- Diseñador UI, encargado de diseñar la interfaz de usuario y proporcionar una experiencia de usuario de calidad
- Modelador 3D, para crear unos modelos 3D iniciales, preparar los elementos gráficos y probar distintas alternativas

1.15.2.3 Fase de desarrollo

- Desarrolladores Flutter, para desarrollar la aplicación Flutter
- Desarrolladores Backend, para implementar los servidores y la API que manejará la interacción con la aplicación y la devolución del modelo 3D
- Programador de gráficos 3D, para optimizar los algoritmos de creación de modelos 3D

1.15.2.4 Fase de Pruebas

- Ingenieros de QA, para probar la aplicación y que cumpla con los requisitos funcionales y de seguridad
- Especialista en pruebas de modelos 3D, para probar la funcionalidad de creación de modelos 3D

1.15.2.5 Fase de implementación

- Ingeniero DevOps, para manejar el despliegue de los servidores

1.15.2.6 Fase de mantenimiento

- Ingenieros de soporte, para proporcionar soporte técnico a los usuarios tras el lanzamiento

Habiendo desglosado toda la intervención humana necesaria en este proyecto, veamos la estimación del coste conforme al salario mensual bruto de cada uno.

Profesional	Salario mensual
Gerente de producto	3500 € / mes
Analista de negocios	2500 € / mes
Diseñador UI	2500 € / mes
Modelador 3D	2500 € / mes
Desarrollador Flutter	3000 € / mes
Desarrollador Backend	3000 € / mes
Programador de gráficos 3D	3500 € / mes
Ingeniero QA	2500 € / mes

Especialista en pruebas de modelos 3D	2800 € / mes
Ingeniero DevOps	3500 € / mes
Ingeniero de soporte	2000 € / mes

Tabla 3 - Coste humano

El total necesario para cubrir los costes humanos, suponiendo una duración total de la implementación y entrega del proyecto de 6 meses, daría un total de:

$$Coste\ humano = (3500\ x\ 3 + 2500\ x\ 4 + \ 3000\ x\ 2 + 2800 + 2000)\ x\ 6 = 187800\ €$$

Sumando este coste humano al total del coste de infraestructura, sería necesario un presupuesto total de 188554.94 €.

1.15.3 Mecanismos de control de calidad

La calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

2 DISEÑO INICIAL

2.1 Especificaciones del sistema

- Captura de imágenes

Uno de los principales requisitos a destacar, la aplicación móvil deberá ser capaz de capturar imágenes, pudiendo acceder a una cámara (disponible en el dispositivo que usa la aplicación) que podrá tomar imágenes con una fluidez

destacada y, de la misma forma guardarlas en el almacenamiento interno del dispositivo para un posterior uso.

- Procesamiento de imágenes

Este TFG debe ser capaz de procesar imágenes tanto desde el lado del servidor como del lado del cliente, para su posterior uso en el modelado.

- Comunicación con el servidor

Para permitir una arquitectura cliente-servidor, debe realizarse con éxito una comunicación a través de una API que conecte el servidor con la aplicación del front.

- Reconstrucción 3D

Al tratarse de una aplicación de modelado tridimensional, deberá ejecutar con éxito el algoritmo de reconstrucción de las imágenes que se han procesado anteriormente.

- Visualización de modelos

Como es lógico, la aplicación deberá permitir una visualización de los modelos generados mediante algún visualizador local u online.

- Exportación de modelos

De la misma forma, los modelos deberán poder exportarse de alguna forma para poder compartirlos o visualizarlos en otro entorno.

- Interfaz de usuario

Se deberá proporcionar una interfaz de usuario en la aplicación con una sencillez que permita a cualquier usuario utilizar la aplicación.

- Gestión de usuarios en la base de datos

En la base de datos en la nube, se deberán gestionar los usuarios presentes en la aplicación, con la posibilidad de realizar acciones en caso de conflicto.

- Rendimiento

Al tratarse de una aplicación con un alto requerimiento hardware, se deberá conseguir una eficiencia aceptable con la infraestructura disponible.

- Usabilidad

Uno de los requisitos más importantes y necesarios, ya que una aplicación usable, conseguirá una mayor retención de usuarios.

- Seguridad

Cuando se trata con aplicaciones que contienen fases de autentificación, deben proporcionar una seguridad destacable para los clientes.

- Compatibilidad multiplataforma

Para llegar a una mayor cantidad de usuarios, se deberá proporcionar una compatibilidad multiplataforma con distintos sistemas.

2.1.1 Requisitos funcionales

Número	Descripción
F1	El usuario debe poder crearse una cuenta
F2	El usuario debe poder iniciar sesión con su cuenta creada
F3	La aplicación debe enviar un correo de verificación y el usuario verificar su cuenta
F4	El usuario debe poder seleccionar imágenes o videos de su galería
F5	El usuario debe poder capturar imágenes o vídeos
F6	El usuario debe poder construir el modelo 3D
F7	El usuario debe poder ver y mover el modelo creado
F8	El usuario debe poder guardar sus modelos localmente
F9	El usuario debe poder ver sus modelos localmente
F10	La aplicación debe poder guardar rutas de los modelos
F11	El usuario debe poder subir modelos a la nube
F12	El usuario debe poder ver modelos que hay ya en la nube
F13	El usuario debe poder cerrar sesión

Tabla 4 - Requisitos funcionales

2.1.2 Requisitos no funcionales

Número	Descripción
NF1	La aplicación debe asegurar una consistencia de todos los elementos en la pantalla (responsive)
NF2	La aplicación debe asegurar la seguridad de los datos del usuario
NF3	La aplicación debe asegurar la comunicación del Backend y del Frontend

NF4	La aplicación debe asegurar su funcionamiento en distintas plataformas
NF5	La aplicación debe ser usable
NF6	La aplicación debe tener unos tiempos de respuesta aceptables para el usuario
NF7	La aplicación debe ser atractiva a la vista con una interfaz de usuario sencilla

Tabla 5 - Requisitos no funcionales

2.1.3 Casos de uso

Crear cuenta	**Id único: F1**
Actor(es)	Usuario / Base de datos
Descripción	Crear una cuenta en la base de datos para el usuario
Evento desencadenado	Inserción de un nuevo documento en Firebase con un usuario y también de una nueva fila en la autentificación
Precondiciones	Tener correo electrónico
Postcondiciones	Enviar un email de verificación
Requerimientos cumplidos	F1
Pasos	
1	Pulsar sobre “Crear cuenta”
2	Introducir correo
3	Introducir contraseña
4	Repetir contraseña
5	Pulsar sobre “Crear cuenta”

Tabla 6 - Caso de uso F1

Iniciar sesión	**Id único: F2**
Actor(es)	Usuario / Base de datos
Descripción	Iniciar sesión en la aplicación

Evento desencadenado	La variable FirebaseAuth.instance.currentuser tendrá ahora almacenado un usuario y se procede al inicio de sesión
Precondiciones	Tener cuenta
Postcondiciones	Entrar a la aplicación
Requerimientos cumplidos	F2
Pasos	
1	Introducir correo correcto
2	Introducir contraseña correcta
3	Pulsar en “Iniciar sesión”

Tabla 7 - Caso de uso F2

Verificación de correo	**Id único: F3**
Actor(es)	Usuario / Base de datos
Descripción	Firebase envía un correo de verificación de cuenta
Evento desencadenado	Cuando el usuario se registra, Firebase procede a enviar una plantilla de correo de verificación
Precondiciones	Haber creado una cuenta
Postcondiciones	Ninguna
Requerimientos cumplidos	F3
Pasos	
1	Crear cuenta
2	Entrar a correo electrónico
3	Seleccionar bandeja de entrada
4	Pulsar sobre el correo de verificación
5	Pulsar en el link para verificar cuenta

Tabla 8 - Caso de uso F3

Seleccionar imágenes o videos de galería	**Id único: F4**
Actor(es)	Usuario / Sistema de archivos
Descripción	Seleccionar un archivo de la galería

Evento desencadenado	La aplicación guardará la ruta hacia ese archivo en una variable para tratarla posteriormente
Precondiciones	Haber dado permiso a la aplicación de acceso a los archivos del teléfono
Postcondiciones	Ninguna
Requerimientos cumplidos	F4
Pasos	
1	Pulsar sobre “Crear Modelo”
2	Seleccionar “Desde galería”
3	Seleccionar las imágenes o video deseado
4	Pulsar sobre “Aceptar”

Tabla 9 - Caso de uso F4

Capturar imágenes o video	**Id único: F5**
Actor(es)	Usuario / Sistema de archivos
Descripción	Toma de instantáneas o videos para poder generar el modelo 3D
Evento desencadenado	La aplicación procesará lo capturado por la cámara para guardarlo en caché del dispositivo
Precondiciones	Ninguna
Postcondiciones	Ninguna
Requerimientos cumplidos	F5
Pasos	
1	Pulsar sobre “Crear modelo”
2	Pulsar sobre grabar
3	Grabar alrededor del objeto deseado
4	Pulsar sobre parar
5	Nombrar el modelo

Tabla 10 - Caso de uso F5

Construir modelo 3D	Id único: F6
Actor(es)	Usuario / Servidor
Descripción	Construir un modelo 3D a partir de un video tomado
Evento desencadenado	La aplicación comenzará el proceso de comunicación con el servidor para la generación del modelo tridimensional
Precondiciones	F5 o F4
Postcondiciones	Tener acceso al servidor
Requerimientos cumplidos	F6
Pasos	
1	Pulsar sobre “Comenzar construcción”
2	Esperar que el servidor genere el modelo
3	Visualizar el modelo

Tabla 11 - Caso de uso F6

Visualizar e interactuar con el modelo	Id único: F7
Actor(es)	Usuario / Interfaz
Descripción	Visualizar modelo 3D e interactuar con el (zoom, moverlo)
Evento desencadenado	Pantalla para visualizar el modelo desde OpenGL en una IP local
Precondiciones	F6
Postcondiciones	Ninguna
Requerimientos cumplidos	F7
Pasos	
1	Crear un modelo
2	Pulsar sobre el modelo
3	Interactuar con él

Tabla 12 - Caso de uso F7

Guardar modelos localmente	Id único: F8
Actor(es)	Usuario / Base de datos
Descripción	Guardar un modelo localmente en el dispositivo
Evento desencadenado	Inserción de una nueva fila en la base de datos local del dispositivo con la ruta hacia el modelo, su nombre y un identificador único
Precondiciones	Tener correo electrónico
Postcondiciones	Ninguna
Requerimientos cumplidos	F8
Pasos	
1	Crear un modelo mediante F6
2	El modelo se guarda automáticamente en el dispositivo

Tabla 13 - Caso de uso F8

Visualizar un modelo localmente	Id único: F9
Actor(es)	Usuario / Base de datos
Descripción	Visualización de un modelo ya creado
Evento desencadenado	La base de datos local proporciona el identificador del modelo para visualizarlo localmente
Precondiciones	Tener algún modelo en la base de datos
Postcondiciones	Ninguna
Requerimientos cumplidos	F9
Pasos	
1	Pulsar sobre el icono de la carpeta
2	Visualizar todos los modelos existentes
3	Pulsar sobre el modelo deseado
4	Visualizar el modelo

Tabla 14 - Caso de uso F9

Guardar ruta del modelo creado	Id único: F10
Actor(es)	Base de datos
Descripción	Guardar ruta del modelo creado
Evento desencadenado	Inserción de una nueva fila en la base de datos local
Precondiciones	Haber creado un modelo
Postcondiciones	Ninguna
Requerimientos cumplidos	F10
Pasos	
1	Crear un modelo
2	Inserción de una nueva fila con la ruta al mismo

Tabla 15 - Caso de uso F10

Subir modelos a la nube	Id único: F11
Actor(es)	Usuario / Base de datos
Descripción	Subir un modelo a la nube para poder visualizarlo posteriormente
Evento desencadenado	Inserción de un nuevo documento en Firebase con un modelo 3D en .glb
Precondiciones	Tener un modelo creado
Postcondiciones	Ninguna
Requerimientos cumplidos	F11
Pasos	
1	Navegar al icono de la carpeta
2	Seleccionar el modelo deseado
3	Pulsar sobre el icono de la nube

Tabla 16 - Caso de uso F11

Visualizar modelos de la nube	Id único: F12
Actor(es)	Usuario / Base de datos
Descripción	Visualizar un modelo previamente subido a la nube por cualquier usuario
Evento desencadenado	Devolución de Firebase del modelo a la aplicación
Precondiciones	Ninguna
Postcondiciones	Ninguna
Requerimientos cumplidos	F12
Pasos	
1	Pulsar sobre el icono de la nube en la barra inferior
2	Pulsar sobre el modelo deseado
3	Visualizar el modelo

Tabla 17 - Caso de uso F12

Cerrar sesión	Id único: F13
Actor(es)	Usuario / Base de datos
Descripción	Cerrar sesión en la aplicación
Evento desencadenado	La variable FirebaseAuth.instance.current user pasará a estar vacía y se volverá a la pantalla de inicio de sesión
Precondiciones	F1
Postcondiciones	Ninguna
Requerimientos cumplidos	F13
Pasos	
1	Navegar a la pantalla principal, con el icono de la casa
2	Pulsar sobre el icono de cierre de sesión arriba a la derecha

Tabla 18 - Caso de uso F13

2.2 Análisis y diseño del sistema

En esta sección se explican los distintos pasos llevados a cabo para el análisis y diseño del sistema. Se dividirá en diferentes secciones, llegando a comprender cómo va a funcionar la aplicación y su comunicación con el Backend.

2.2.1 Diseño de la infraestructura del sistema

Al tratarse de un TFG con un diseño completo de aplicación, con un servidor y con una interfaz de Frontend, se debe aportar una infraestructura adecuada a la situación, con un servidor con las siguientes características:

- Gráfica 1650 GTX NVIDIA
- 32 Gb RAM
- Procesador Intel i7 10ª generación
- SSD 500 Gb

Este servidor será el encargado del procesamiento de imágenes y el envío de los modelos a la aplicación. Será diseñado mediante Flask. Es un framework escrito en Python para el desarrollo de la API de conexión del servidor con la aplicación basado en WSGI. Flask proporciona una gran variedad de posibles acciones entre las que encontramos HTTP GET, POST, PUT, DELETE etc… La aplicación flask se ejecutará en todas las interfaces disponibles y contará con los siguientes métodos:

- Allowed_file(filename), contiene las posibles extensiones de archivos que el servidor puede recibir, de esta forma evitamos ataques que comprometan la seguridad del servidor
- Upload_images(), método POST que utiliza la aplicación para subir imágenes al servidor
- Upload_video(), método POST que utiliza la aplicación para subir vídeo al servidor
- Load_data(), método POST para la división del vídeo en imágenes utilizando nerfstudio y su método *ns-proccess-data,* este método generará varias carpetas de imágenes junto con un archivo transform.json que contiene la información para entrenar el modelo

- Train(), método POST para el entrenamiento del modelo utilizando las imágenes previamente creadas con nerfstudio y su método *train nerfacto,* este método genera un archivo config.yml con la configuración necesaria así como las respectivas matrices de transformación, la calibración de la cámara y las coincidencias encontradas
- Generate_model(), método POST para la creación del modelo generando un archivo .obj junto con dos archivos que contienen el material utilizado para la textura mediante el comando de nerfstudio *ns-expost poisson*
- Return_model(), método POST que se encarga de transformar el archivo .obj a .glb junto con la información necesaria (texturas) para devolverlo a la aplicación y poder visualizarlo

Igualmente, es necesario utilizar dos bases de datos diferentes, una local, con SQL para la gestión de modelos locales, y otra en la nube, con Firebase, para la gestión de usuarios y de modelos en la nube. El diseño de la infraestructura completa se muestra a continuación.

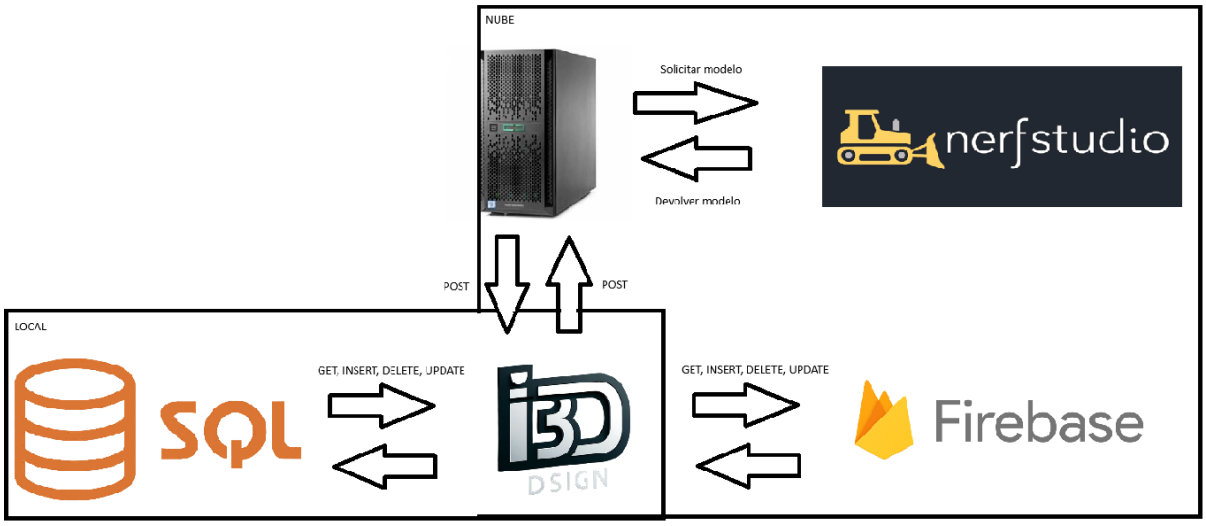


Ilustración 5 - Diseño de la infraestructura

2.2.2 Diseño de las bases de datos

Como se puede observar en el esquema anterior, existen 2 bases de datos distintas que tiene diferentes funciones, en primer lugar, Firebase, encargada de la

gestión de usuarios y de modelos en la nube, y, en segundo lugar, SQL, encargada de la gestión de modelos locales.

2.2.2.1 Firebase

Firebase, proporcionado por Google, es más que un gestor de bases de datos, permite, a parte de la gestión de una base de datos de diferentes formas:

- Gestión de usuarios mediante un apartado dedicado únicamente a autentificación
- Gestión de archivos
- Gestión de bases de datos a tiempo real
- Monitorización de la aplicación
- Visualización de usuarios en tiempo real
- Notificaciones push
- Google Analytics
- Admob

Ilustración 6 - Firebase Authentication

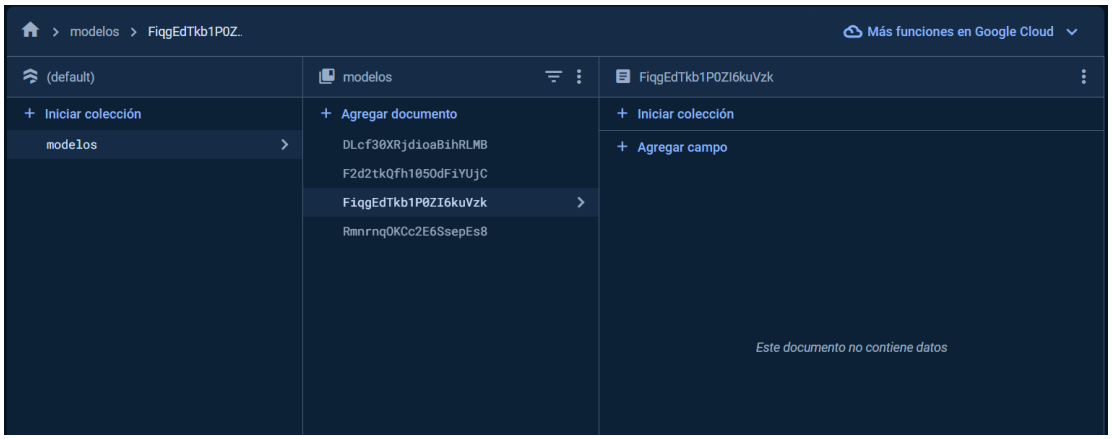


Ilustración 7 - Firebase Firestore

Ilustración 8 - Firebase

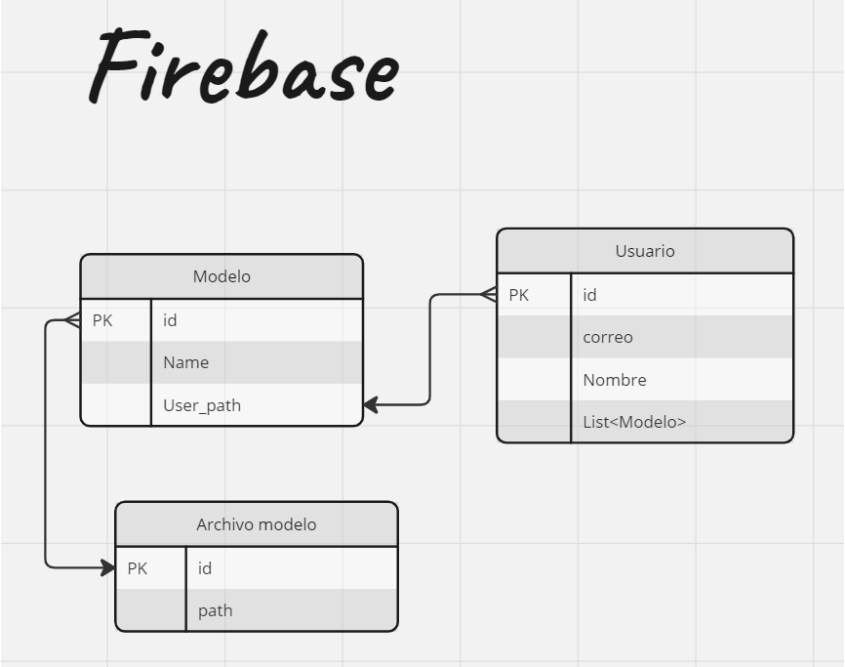


Ilustración 9 - Diagrama de Firebase

2.2.2.2 SQL

Se trata de una base de datos en local muy sencilla, encargada de gestionar las rutas de los modelos junto con sus nombres, también hay una clase encargada de gestionar esta base de datos, llamada DatabaseHelper, que a su vez está gestionada por una clase con un modelo provider consumer. El modelo provider consumer es muy popular dentro del lenguaje de programación Dart y Flutter, proporciona la capacidad de dar a una clase el conocido patrón observador, gestionada mediante un Change Notifyer. En cada método que modifique algún atributo de esta clase, se indica al final un NofityListeners(), este método avisa a aquellas clases consumidoras (Consumer<Modelo>) de que el dato que están usando ha cambiado.

La base de datos local contiene las siguientes funciones:

- Insertar filas - INSERT
- Modificar filas - UPDATE
- Borrar filas – DELETE
- Consultar filas – QUERY ALL ROWS

Dentro de la clase, se encuentra un mapa para gestionar de la manera más eficiente posible los datos almacenados y poder buscar de la forma más rápida posible.

Ilustración 10 - Diagrama SQL

2.2.3 Diagrama de clases

Para previsualizar de una manera sencilla cómo se estructura la aplicación, veremos un diagrama de clases que lo resume con sus atributos y funciones.

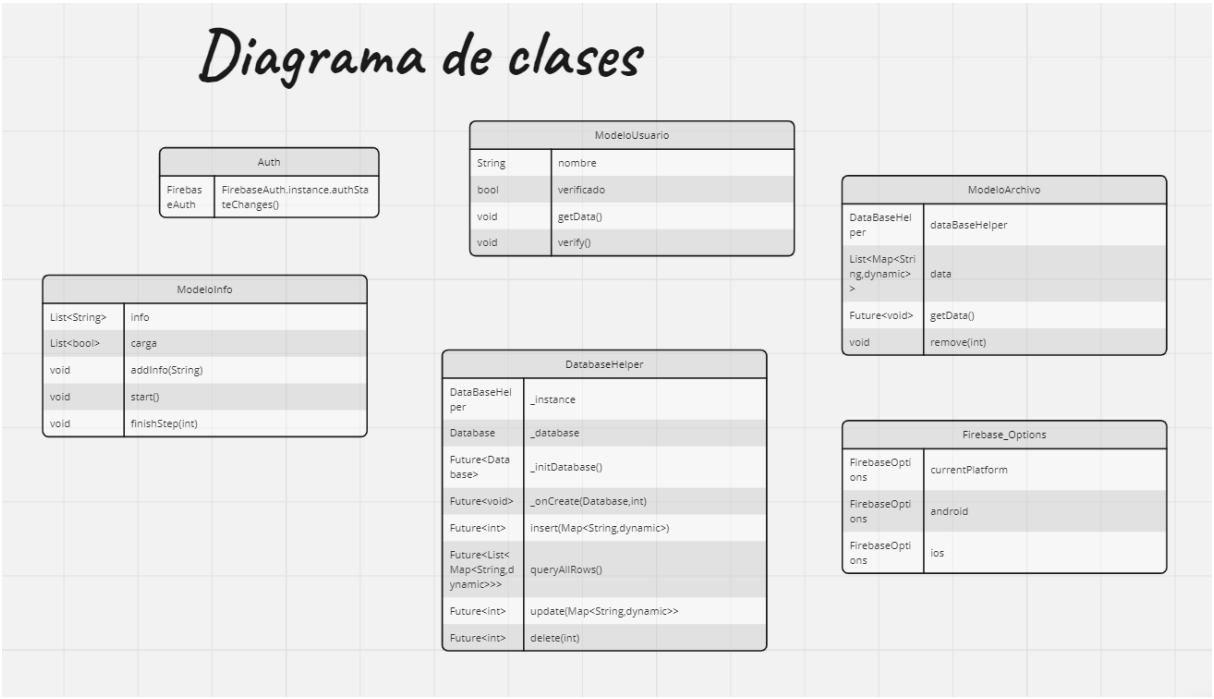
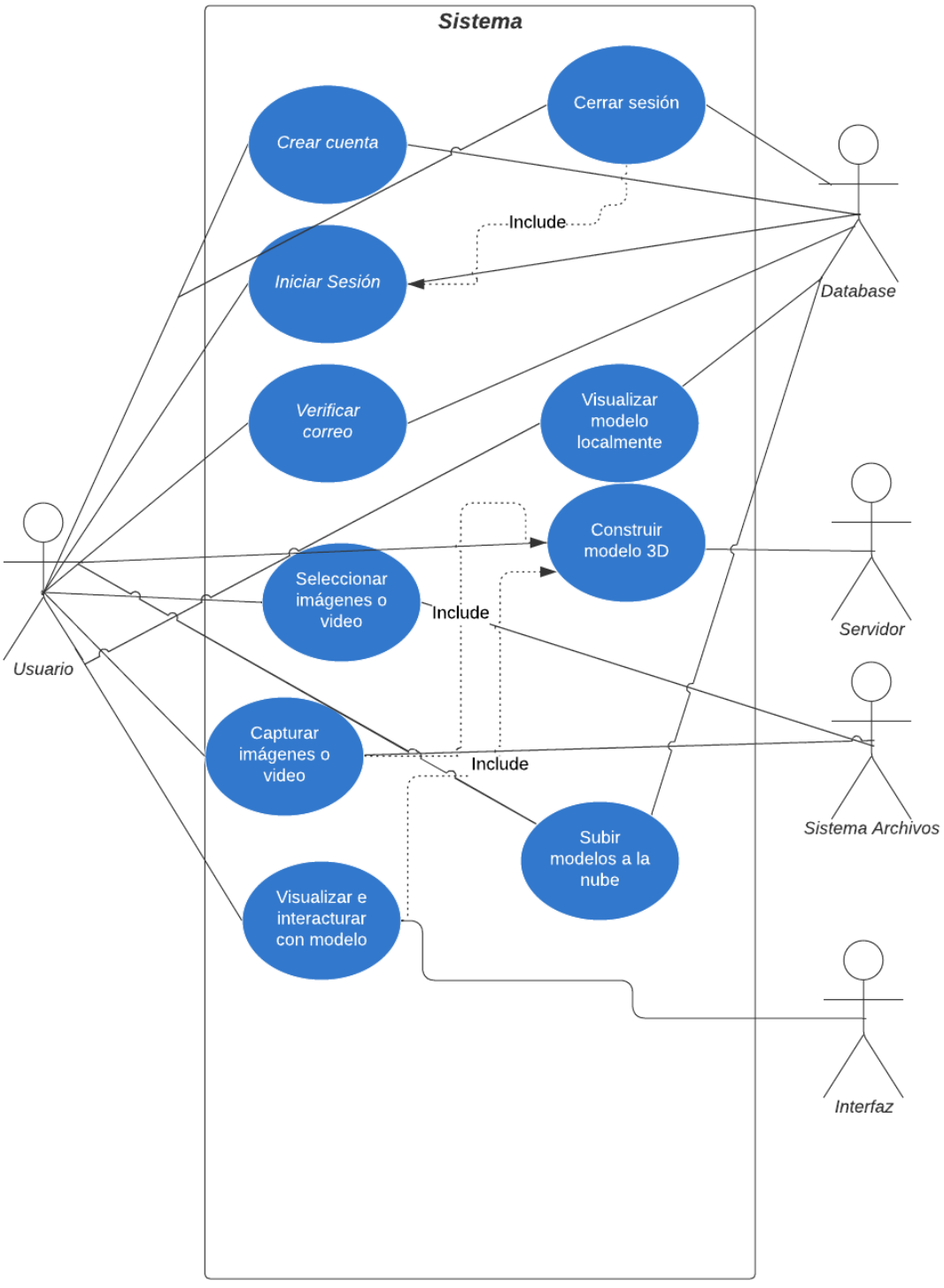


Ilustración 11 - Diagrama de clases

2.2.4 Diagrama de casos de uso



2.2.5 Diagramas de secuencia

Los diagramas de secuencia a continuación presentan los casos de uso anteriores.

2.2.5.1 Login (F1)

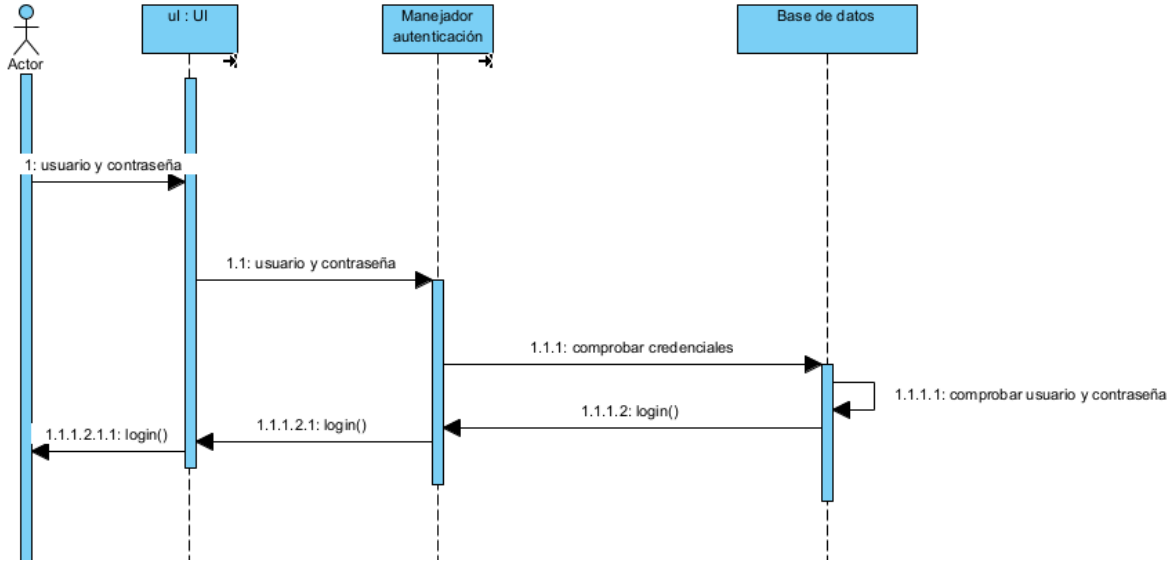


Ilustración 12 - Diagrama de secuencia F1

2.2.5.2 Crear cuenta y verificar (F2 y F3)

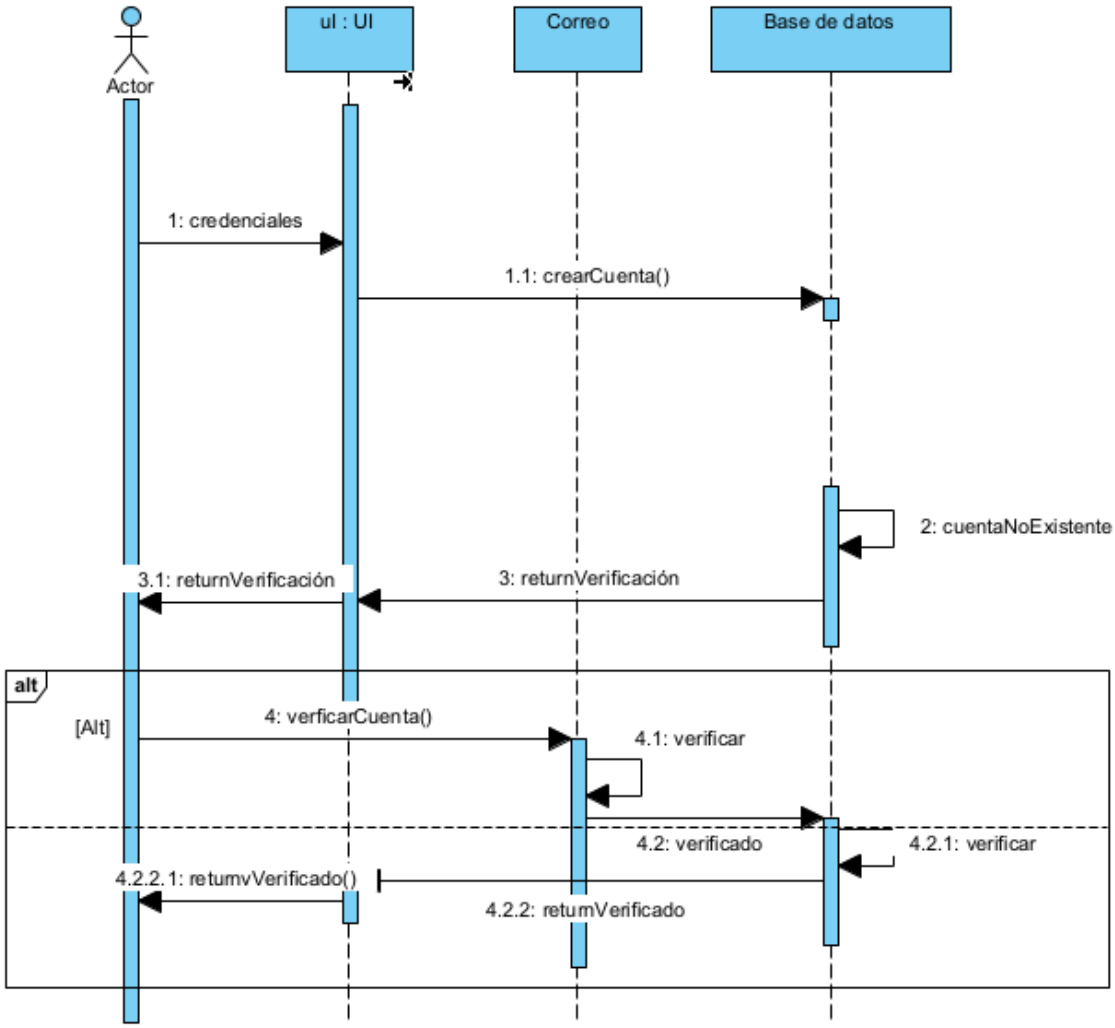


Ilustración 13 - Diagrama de secuencia F2 y F3

2.2.5.3 Crear modelo, guardar modelo, ver modelo, capturar video y seleccionar archivo (F4, F5, F6, F7 y F8)

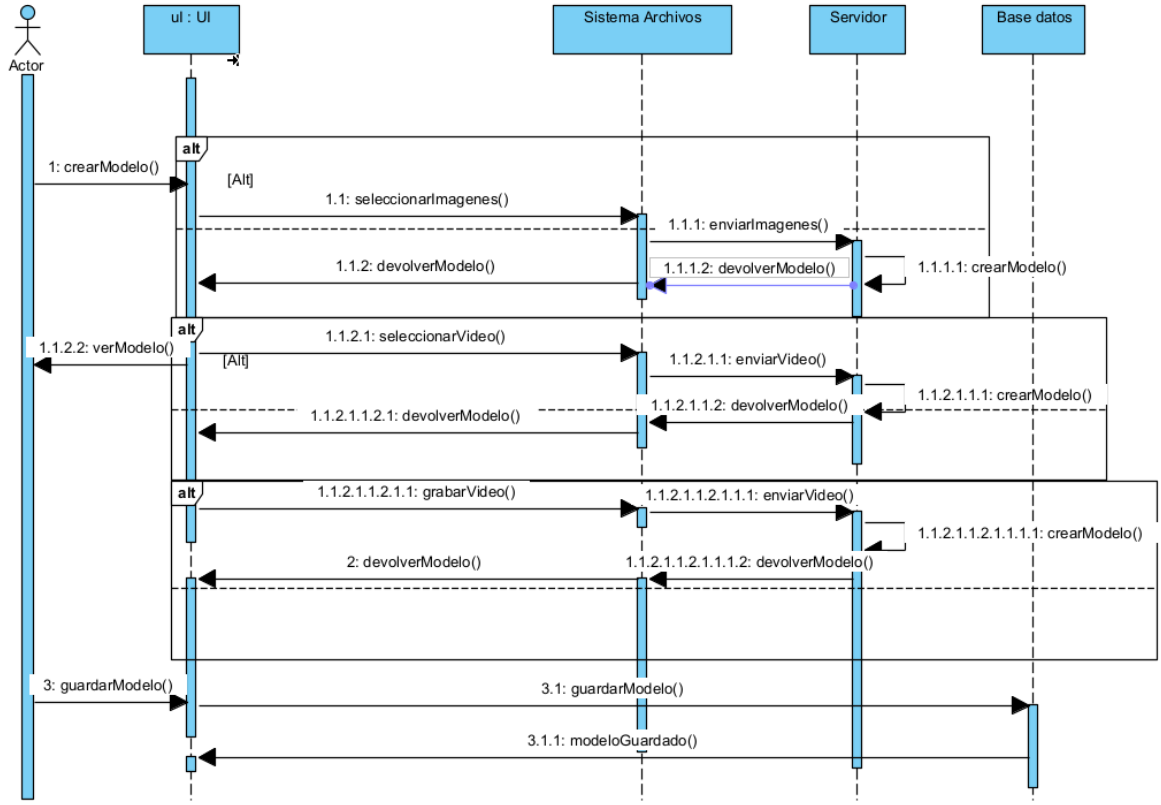


Ilustración 14 - Diagrama de secuencia F4, F5, F6, F7 y F8

2.2.5.4 Subir modelos a nube, visualizar modelos de la nube y cerrar sesión (F11, F12 y F13)

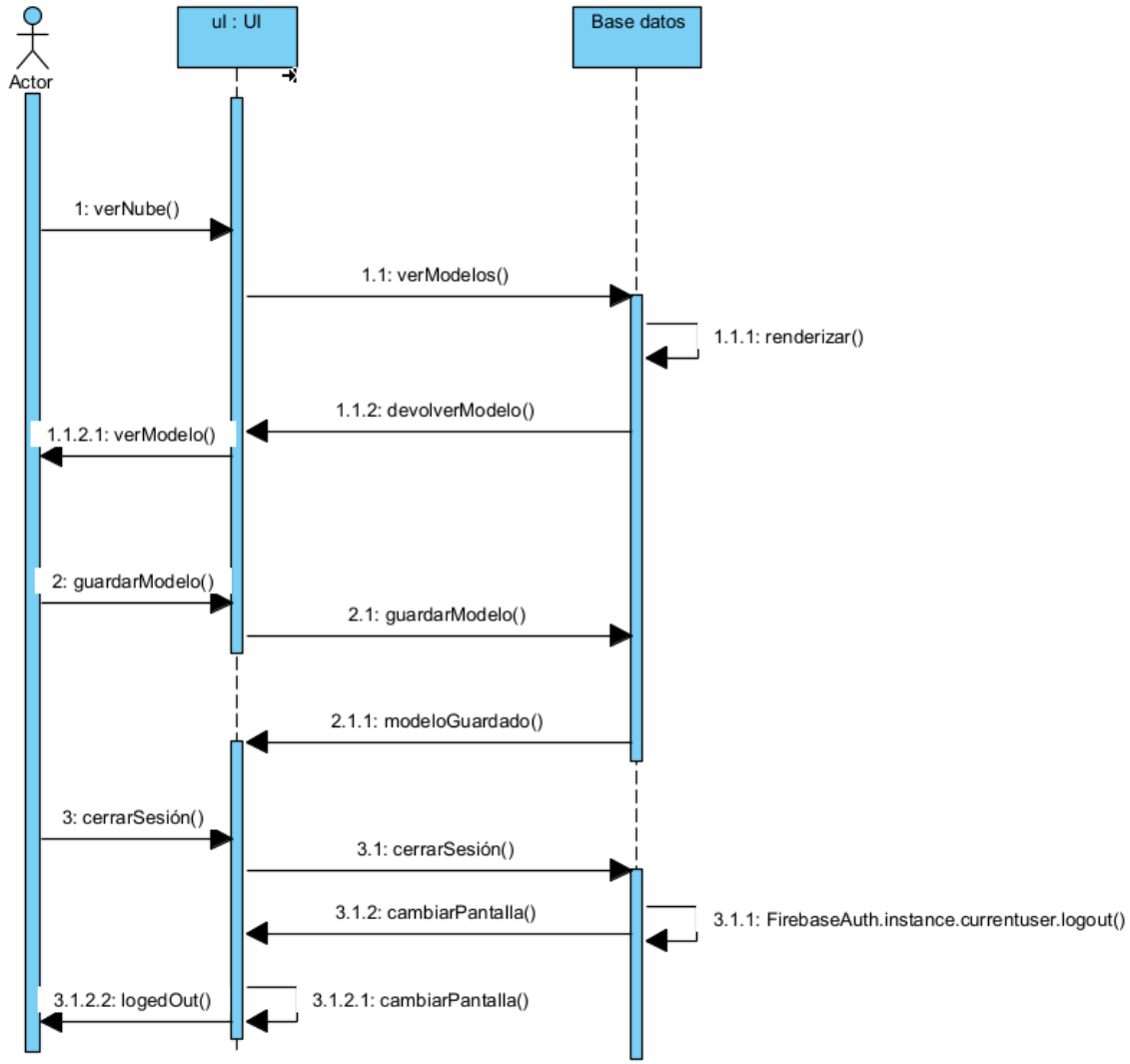


Ilustración 15 - Diagrama de secuencia F11, F12 y F13

2.2.6 Diseño de la interfaz y storyboards

El diseño de la interfaz es de las partes cruciales que conforman el análisis y diseño de un proyecto de aplicación móvil, ya que, a través de ella los usuarios podrán realizar las acciones que quieran. Para poder realizar con facilidad la interfaz, se ha utilizado Figma para realizar bocetos y el storyboard de la aplicación, a continuación se pueden ver las imágenes.

Las dos primeras imágenes corresponden a la parte inicial de la aplicación, con su pantalla de inicio y la pantalla de login.

Ilustración 16 - Storyboard Inicio y Login

La siguiente, la pantalla principal.

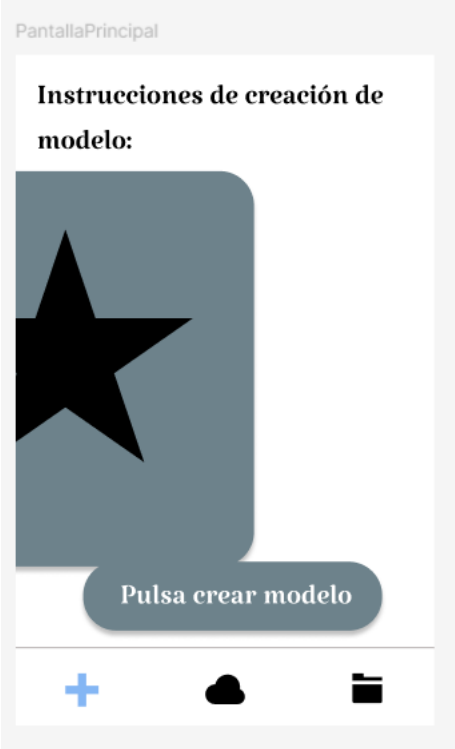


Ilustración 17 - Storyboard Pantalla Principal

De la misma forma, podría navegar a “Mis modelos”.

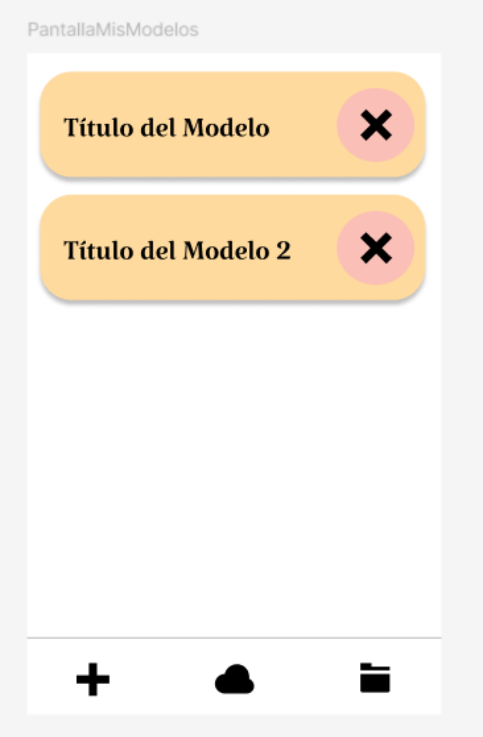


Ilustración 18 - Storyboard Pantalla Mis Modelos

Y, por último, a la pantalla de Cloud.

Ilustración 19 - Storyboard Cloud

Es importante recordar que esto solo se trata de un boceto, por lo que, no contendrá con exactitud las pantallas finales de la aplicación, ni contiene las pantallas de carga, las de la cámara o visualizar modelos.

También, como presentan las primeras dos fotos, se diseñó un logo para la aplicación, creado por IA con las pautas marcadas, la entrada para que la IA Dall-e generase es logo fue:

"Diseña un logo para una aplicación de creación de modelos 3D, con nombre I3D Design, marcando bordes, reduciendo "Design" por Dsign, con dos colores (blanco y azul oscuro) y aprovechando el principio de la D para el 3 y el punto de la I"

Si se quisiese comercializar la aplicación, no habría problemas con el logo ya que está libre de derechos, permitiendo su lanzamiento a las tiendas de aplicaciones.

Ilustración 20 - Logo I3D

3 DESARROLLO

Para el desarrollo del TFG se ha utilizado una metodología de desarrollo ágil, por iteraciones. Para el desarrollo se numerarán las iteraciones por periodos de 14 días, de las siguiente forma DiaInicialDiaFinal-MesFinal-NumeroIteración, siendo un ejemplo de ello 1428-01-IT1, para el 14 de enero al 28 de enero, con iteración 1.

3.1 1529-01-IT1

En esta primera iteración se marca la puesta en marcha del servidor y de la aplicación Flask, junto con la instalación de Nerfstudio en el servidor con sus respectivos paquetes necesarios.

- Instalación del servidor
- Apertura del puerto 8000 para poder acceder por HTTP
- Instalación de Nerfstudio
- Configuración de entorno con Anaconda

En primer lugar, el proyecto necesita un Backend robusto, por lo que es necesario instalar de manera correcta todo lo necesario, comenzando por la aplicación Flask. Para realizar todo ello de una manera ordenada se crea un entorno con anaconda llamado “nerfstudio”, para poder acceder al mismo es necesario ejecutar el comando “conda actívate nerfstudio”, previamente habiéndolo creado con “conda créate nerfstudio”. El servidor ejecuta el sistema operativo Windows 11.

Continuando con la configuración, se instala Nerfstudio en el entorno, teniendo especial cuidado con la versión de los paquetes, ya que, en la próxima iteración hubo que arreglar errores con los mismos.

Es necesario abrir el puerto 8000 (donde se ejecutará la aplicación) para poder acceder al servidor mediante HTTP sin que haya problemas. No es obligatorio la apertura de ese puerto, también se permite cualquier otro.

Con todo esto configurado, se procede a la creación de una carpeta en el servidor que gestionará todo lo relativo al mismo.

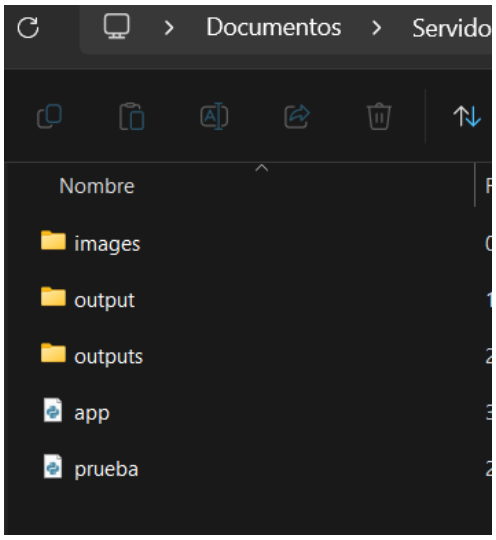


Ilustración 21 - Carpeta raíz del servidor

La carpeta raíz se divide en varias carpetas:

- Images: contiene todos los archivos que los usuarios subirán
- Output y outputs: salidas que genera Nerfstudio
- App: Aplicación flask que se ejecuta en el puerto 8000
- Prueba: Aplicación de pruebas

3.2 2912-02-IT2

La segunda iteración del proyecto gestionó una serie de errores encontrados, así como el comienzo de la programación de la aplicación con Android Studio.

- Implementación de la pantalla inicial y el login (sin el backend)
- Corrección de errores en el servidor (dependencias de Nerfstudio)
- Inserción del logo en pantallas iniciales

La implementación de la pantalla del login y la inicial es algo importante ya que el usuario es lo primero que encuentra, y, se diseña de una forma atractiva, con botones grandes y pocas alternativas para no llevar a error.

El servidor experimentó una serie de errores con las dependencias de Nerfstudio, en concreto con CUDA y PyTorch, estos paquetes contienen librerías de aprendizaje profundo y, si no se instalan las versiones adecuadas para ambos,

generan incompatibilidades que evitan la ejecución de las mismas, pudiendo arreglarlo con varias soluciones encontradas en GitHub.

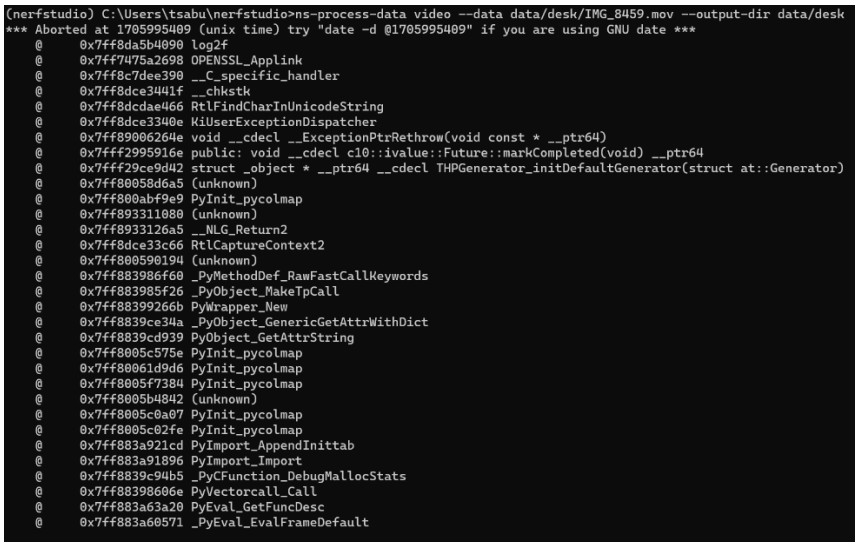


Ilustración 22 - Error nerfstudio GitHub

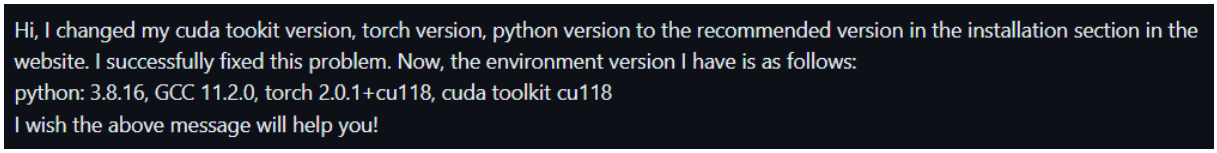


Ilustración 23 - Solución error versiones

3.3 1224-02-IT3

Iteración de comienzo de pruebas de nerfstudio en local, paralelamente puesta en marcha de Firebase y continuación de la aplicación con Android Studio.

- Pruebas en local de nubes de puntos
- Programación de las tres primeras primitivas de Flask
- Creación de la pantalla principal de la aplicación
- Creación del proyecto Firebase con id tfginformática -ffc26 y nombre I3Dsign
- Conexión de la aplicación con Firebase

Para la prueba de generación de modelos 3D, se ejecutó en la máquina local nerfstudio, generando como resultado una buena nube de puntos de ejemplo.

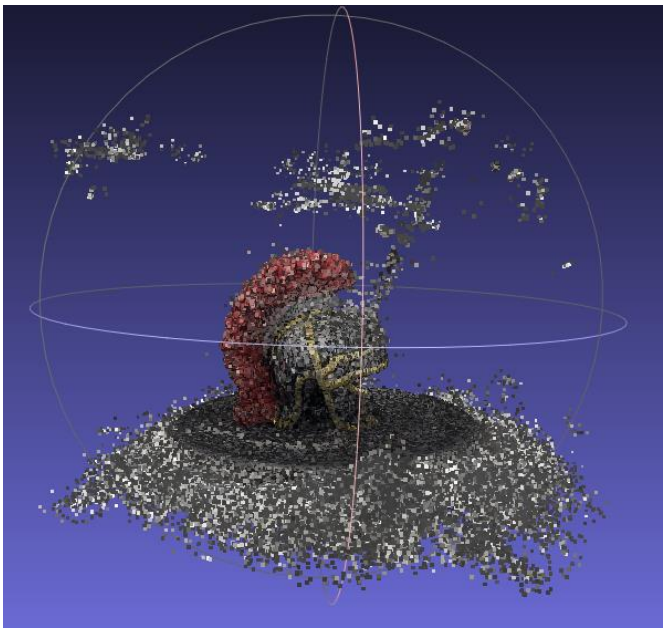


Ilustración 24 - Nube de puntos

Las primeras funciones programadas en Flask son las de captura de imágenes, video, y tratamiento del video para su descomposición en imágenes o Frames. Estas funciones, están en el servidor con los nombres:

- Upload_images(), con el @app.route /upload-images
- Upload-video(), con el @app.route /upload-video
- Load_data(), con el @app.route /load-data

Firebase proporciona una sencilla interfaz para su puesta en marcha y una forma fácil de conexión de Flutter con la plataforma, mediante Firebase CLI que conecta nuestro proyecto local con nuestro proyecto en la nube, generando un archivo en la carpeta “lib” llamado firebase_options.dart.

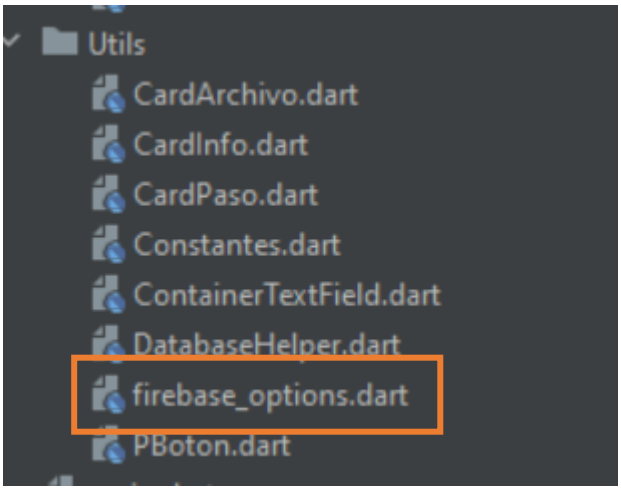


Ilustración 25 - Archivo firebase_options

Junto con la pantalla principal, se crea un archivo Constantes.dart que contendrá las constantes del proyecto, tanto para las instrucciones, como colores personalizados en hexadecimal o rutas a assets.

3.4 2409-03-IT4

Iteración principalmente de programación:

- Programación de la cámara y captura de video
- Conexión con el servidor inicial para envío de video
- Programación de provider-consumer para las etapas de modelado con el servidor

Para la creación de la pantalla de la cámara, es necesario destacar el uso del paquete camera de Flutter, permitiendo acceder a las cámaras frontal y trasera del dispositivo para la toma de video.

La conexión con el servidor se realiza con el paquete http, creando un método inicial que envía los datos parseados en json a el servidor por medio de la siguiente url:

http://ipservidor:8000/upload-video

De esta manera, se programa también el provider-consumer que gestionará la información que el usuario ve cuando el servidor está trabajando en el modelado, mediante la clase ModeloInfo.dart, y, añadiendo a el archivo main el ChangeNotifierProvider correspondiente.

3.5 0923-03-IT5

Algún arreglo de errores, continuación de programación del servidor para la generación de modelos y programación de nuevas vistas.

- Arreglo de error de conexión al necesitar permitir comunicación desde el archivo manifest de Android Studio
- Programación del método de entrenamiento del modelo
- Programación de la vista “Mis modelos”
- Creación de la base de datos SQL

Tras ver que la conexión de la aplicación con el servidor no se establecia de una forma adecuada, es necesario añadir varias líneas al archivo manifest para permitir la conexión.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
```

Ilustración 26 - Archivo Manifest

Como se observa en la imagen, es necesario permitir el acceso a internet del dispositivo, de lo contrario, no se obtendrá una conexión a internet desde la aplicación.

El método de entrenamiento de la aplicación Flask tiene el nombre /train del @app.route y se encarga de entrenar el modelo para su posterior generación.

La vista modelos contiene una lista de modelos, cuyas rutas están almacenadas en la base de datos creada con el archivo DatabaseHelper, siendo necesario incluir un nuevo provider-consumer para la misma.

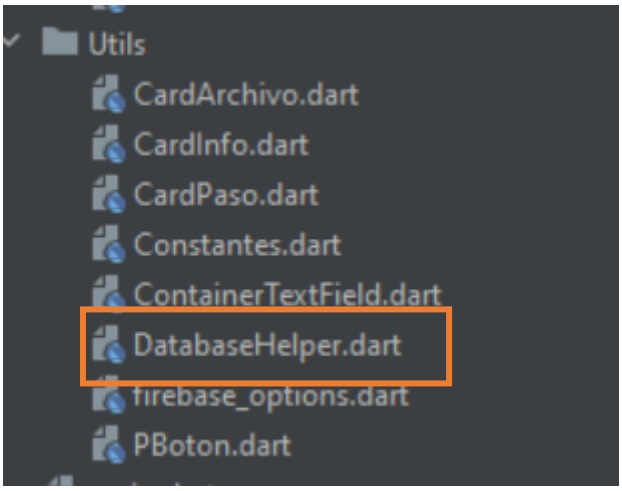


Ilustración 27 - Archivo DatabaseHelper

3.6 2306-04-IT6

- Lógica de inicio de sesión con Auth
- Creación de usuarios y verificación de correo
- Pruebas con el servidor y la aplicación

Para la lógica de inicio de sesión se lleva a cabo con una variable que contiene Firebase, FirebaseAuth.instance.currentuser ¡= null, es decir, si algún usuario ha iniciado sesión, el StreamBuilder ejecutará el código para mostrar la pantalla principal de la aplicación, en caso contrario, ejecutará el código para mostrar la pantalla de login.

Es necesario para que esta lógica funcione, que el usuario pueda crear una cuenta y registrarse en la aplicación, para ello existe otra pantalla que le permite crear una cuenta con un correo y una contraseña, es importante destacar que debe hacerlo con un correo válido, ya que, tras introducir los datos, se enviará un correo de verificación mediante FirebaseAuth.instance.currentuser.sendverificationemail(). Este correo de verificación contiene la siguiente plantilla.

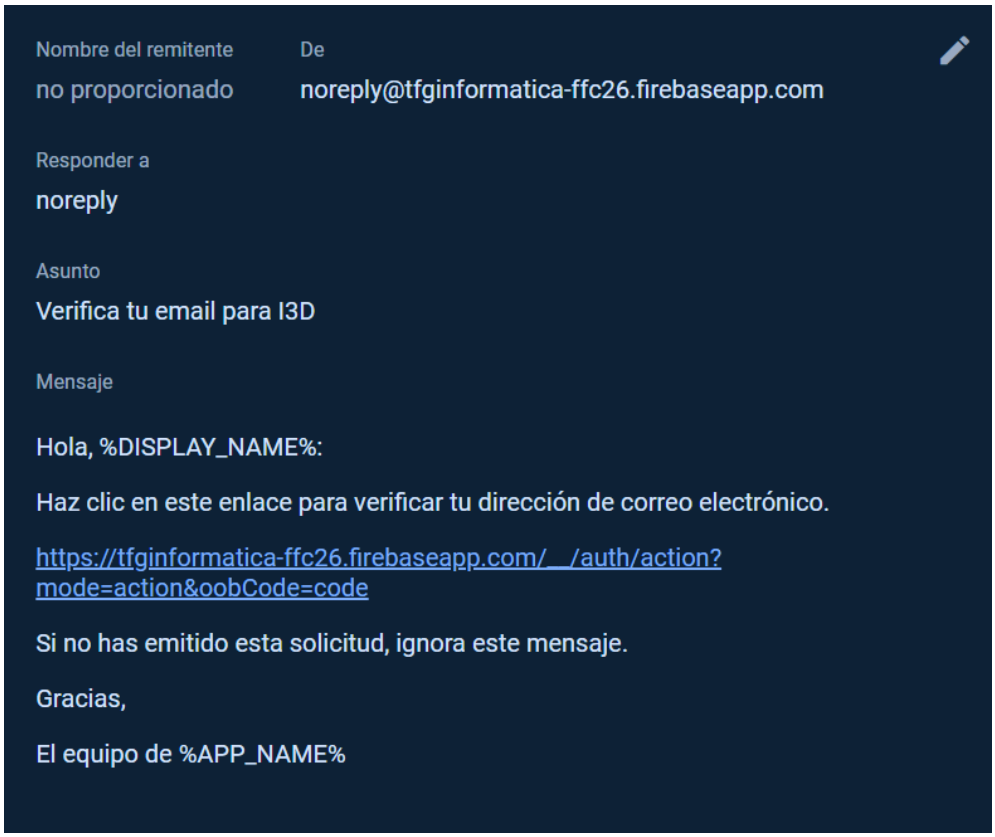


Ilustración 28 - Plantilla correo de verificación

Firebase proporciona la posibilidad de enviar correos de verificación, de cambio de contraseña o de notificaciones.

En esta iteración también se prueban las conexiones ya arregladas con el servidor, pudiendo enviar video y observando como se crea en una carpeta con el correo del usuario que ha iniciado sesión, teniendo en su interior el video y su descomposición en imágenes.

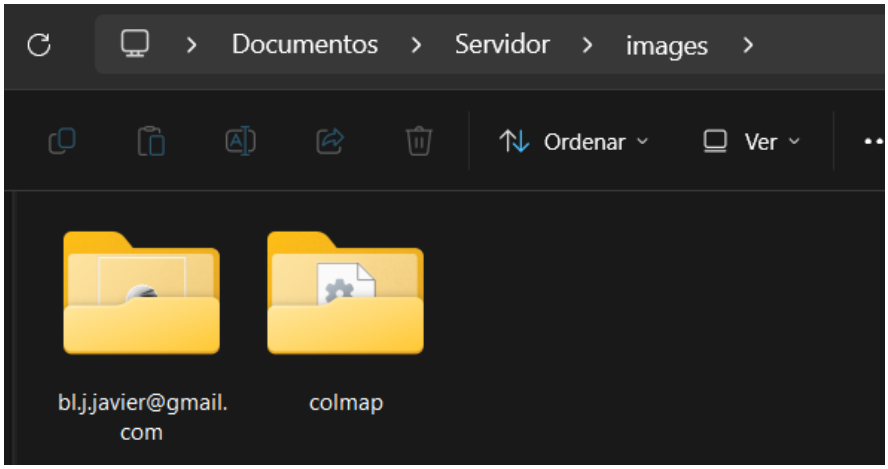


Ilustración 29 - Carpeta Servidor con correo

3.7 0620-04-IT7

- Envío de video y entrenamiento de modelo
- Provider-consumer de la información del servidor arreglado

Se comienzan las pruebas enviando videos y observando como el servidor comienza a entrenar el modelo una vez se ha cargado el video y generado las imágenes. Esto funciona debido a que, cada vez que termina de ejecutar los comandos necesarios el servidor, envía de vuelta una respuesta POST a la aplicación indicando:

- Código 200 – se ejecutó con éxito
- Código 500 – hubo algún problema

Una vez esto está en la aplicación, se muestra algún error si fuese el caso, o, se continúa con el siguiente paso, enviando una nueva solicitud POST al servidor. El provider consumer tenía un error en la iteración previa, al comenzar de nuevo, no se borraban los datos antiguos, por lo que el usuario veía varias veces el mismo mensaje, se arregla eliminando los datos de la lista una vez se empieza de nuevo.

3.8 2004-05-IT8

- Generación del modelo en el servidor, la función /generateModel
- Corrección de errores en Auth

Se añade una nueva función para la generación del modelo 3D, tras haber entrenado el mismo, esta genera el modelo con la información predeterminada.

La lógica Auth no permitía iniciar sesión correctamente, por lo que se debe cambiar la pantalla inicial al login, evitando problemas al recargar.

3.9 0418-05-IT9

- Configuración del Firestore de firebase
- Programación de widgets personales

Como se ha mencionado en otros apartados, Firebase proporciona un gestor de bases de datos llamado Firestore, que va a parte del Authenticator, este da la

posibilidad de crear una base de datos NoSQL donde gestionar datos. En lugar de trabajar con tablas, funciona mediante documentos, pero es la misma filosofía.

Para dar un toque personal a la aplicación se crean widgets personales con colores seleccionados cuidadosamente, estos son colores pastel que, psicológicamente, son mas atractivos al usuario. Los códigos hexadecimales de los colores son:

- Color primario: 0xFF415A66
- Color secundario: 0XFF6D828B

Los dos primeros caracteres, es decir, FF, determinan la transparencia del color, siendo FF opaco completamente.

Color primario	Color secundario

Tabla 19 - Colores personalizados

3.10 1801-06-IT10

- Programación de la vista de modelos
- Corrección de tamaño de widgets en pantalla principal y mis modelos
- Añadir nueva fuente Oswald
- Devolver modelo del servidor

Para poder visualizar los modelos, se utiliza la librería flutter_3d_controller, que permite, usando OpenGL, una visualización online desde una ip local (127.0.0.1) el modelo 3D.

Se corrige el tamaño de la información en pantalla de la principal y el ancho de los Card en mis modelos.

Es importante destacar que la fuente por defecto de flutter es muy pobre, por lo que se busca una nueva fuente en Google Fonts y se selecciona Oswald.

Por último, se programa el método post para devolver el modelo generado a la aplicación, con el @app.route /returnModel

3.11 0115-06-IT11

- Programación de vista Cloud
- Optimización de código en el servidor
- Programación de conversor de obj a glb

La vista cloud es muy sencilla, permite visualizar modelos que encontramos en el servidor, con una lógica parecida a la pantalla mis modelos.

La optimización de código es algo que no es tedioso ya que se programa en las distintas iteraciones con mucho cuidado y atendiendo a lo que el usuario quiere.

Ya que la librería de visualización de modelos no permite archivos .obj (generados por Nerfstudio), se debe convertir éste antes de enviarlo de vuelta, por lo tanto, se convierte previamente, se guarda en la misma carpeta que el modelo .obj, y se manda de vuelta a la aplicación, sin necesidad de enviar archivos de textura ya que el propio glb contiene la información.

3.12 Pruebas finales

3.12.1 Pruebas de verificación del sistema

Al haber realizado varias pruebas de generación de modelos, es importante destacar la calidad sorprendente de los modelos generados, con unas texturas muy buenas, sombreadas y con bordes notados. En cuanto a la gestión de usuarios, se crearon varias cuentas y se podían conectar simultáneamente sin problemas encontrados. El servidor, al no tener altas prestaciones, no permitía generar varios modelos al mismo tiempo, pero, este problema se puede solventar con una infraestructura mejor.

3.12.2 Validación de la usabilidad del sistema

Se comprobó la usabilidad con dos usuarios, los cuales no tuvieron ningún problema, al tratarse de una interfaz muy simple, con pasos detallados, es de notable importancia la facilidad con la que supieron como llegar a cada parte de la aplicación. De la misma forma, supieron crear su propia cuenta e iniciar sesión sin problemas.

4 MODELO DE NEGOCIO

Es importante tener un plan de negocio de cara al lanzamiento de aplicaciones móviles donde el mercado es muy amplio y ofrece mucha oportunidad de negocio. Actualmente, según datos de AppBrain, existen más de tres millones de aplicaciones disponibles en Play Store y cerca de dos millones en App Store. Este primer dato debe servir como punto de partida para analizar el número de usuarios en cada plataforma, claramente mayor en Android.

4.1 Objetivos

4.1.1 Objetivos cualitativos

Id	Descripción
C1	Explotar un nicho en auge
C2	Ofrecer una experiencia de usuario de calidad
C3	Conseguir recurrencia de usuarios
C4	Actualizaciones periódicas

Tabla 20 - Objetivos cualitativos

Los objetivos cualitativos definen aquellas tareas que deben cubrirse para proporcionar una calidad de notable nivel al proyecto en cuestión.

4.1.2 Objetivos cuantitativos

Id	Descripción
CN1	Abarcar el mayor número posible de plataformas
CN2	Conseguir tiempos de respuesta mínimos
CN3	Reducir costes de infraestructura con alta calidad
CN4	Permitir el uso para todos los públicos

Tabla 21 - Objetivos cuantitativos

Los objetivos cuantitativos permiten definir los usuarios alcanzados, así como el beneficio obtenido.

4.2 Análisis del entorno

Un proceso previo al lanzamiento de una aplicación al mercado es el estudio del mismo. Como se ha citado en la sección estado del arte, existen diferentes aplicaciones en las tiendas que permiten la creación de modelos tridimensionales. Por lo tanto, teniendo en cuenta lo anteriormente mencionado:

- La mayoría no poseen un sistema de autenticación
- Interfaz de usuario complicada
- En algunas, tiempos de respuesta elevados

Se debe realizar un análisis DAFO para plasmar de una forma gráfica la oportunidad de negocio.

Ilustración 30 - Análisis DAFO

Como se observa en la sección amenazas, las mayoría de aplicaciones poseen una infraestructura muy buena para conseguir tiempos de respuesta, por lo que, el primer punto a tener en cuenta para invertir el presupuesto existente sería en un servidor potente con características de alto nivel. Conforme se obtenga un beneficio de la aplicación, se deberá seguir invirtiendo en este ámbito hasta conseguir mejorar a la competencia.

Al tratarse de un mercado relativamente nuevo, se tiene la oportunidad de captar a mayor número de usuarios. Viendo este punto, se concluye que es necesario lanzar la aplicación tanto en Play Store como en App Store, para permitir su uso tanto a usuarios de Android como de Apple, pudiendo lanzarlo en un futuro para otros sistemas con menos usuarios como tiendas de Samsung o incluso adaptar la aplicación para poder utilizarla en SmartTVs.

4.3 Plan de mercadotecnia

En esta sección se analizará cómo la aplicación puede generar ingresos de la forma más óptima posible.

Existen multitud de posibilidades para monetizar una aplicación móvil entre las que encontramos:

Tipo	Descripción
Pago único	Único pago en la Store con un precio fijo
Compras In-App	Micro transacciones por acceder a funcionalidades de la aplicación
Modelo Freemium	Aplicación gratis con algunas funcionalidades capadas, con posibilidad de alcanzarlas con pago único
Suscripciones	Pago de una tarifa (por ejemplo mensual) por el acceso completo a la aplicación
Publicidad	Anuncios en la aplicación
Afiliación	Productos de terceros ganando por comisión
Venta de datos	Venta de datos de los usuarios
Modelo híbrido	Modelo freemium junto con publicitario

Tabla 22 - Tipos de monetización

Ahora, analicemos la viabilidad de las diferentes formas de monetización con el proyecto actual:

Tipo	Viabilidad	Explicación
Pago único	No viable	No es suficiente para cubrir gastos
Compras In-App	No viable	No se dispone de muchas funcionalidades para poder realizar micro transacciones
Modelo Freemium	Viable	Deben marcarse con cuidado las partes premium
Suscripciones	Viable	Se puede fijar un precio trimestral bajo, con muchos usuarios se podría conseguir un alto beneficio
Publicidad	Viable	Buena opción para insertar anuncios antes de crear el modelo o de subirlo a la nube
Afiliación	No viable	No se disponen de productos de terceros
Venta de datos	No viable	Muchos problemas legales
Modelo híbrido	Viable	Presenta dos modelos viables, si se realiza con cuidado, buena alternativa

Tabla 23 - Viabilidad de monetización

Observando las diferentes alternativas, se concluye que la mejor alternativa a aplicar para obtener beneficio de la aplicación sería un modelo híbrido. Este modelo debe realizarse con cuidado, teniendo en cuenta los siguientes puntos:

- El modelo premium no debe ser muy caro
- Los anuncios no deben ser demasiado intrusivos ni recurrentes, evitando que el usuario se canse de visualizar tantos anuncios
- Se deben marcar puntos estratégicos para visualizar los anuncios, por ejemplo, mientras el servidor está realizando el modelo
- Al ser una aplicación con pocas funcionalidades, pero muy marcadas, se debe utilizar en pocas de ellas el modelo premium, pero creando una necesidad a partir de las demás, por ejemplo, permitiendo crear hasta un máximo de 10 modelos al mes, tras crear 10 se debe pagar una tasa trimestral de 35 €

Después de observar el modelo de negocio, los cálculos de facturación serían los siguientes.

$$Beneficio = Nº\ suscripciones\ trimestrales\ x\ precioSuscripcion + Nº\ anuncios\ visualizados\ x\ beneficio\ por\ anuncio$$

Sabiendo que los beneficios de los anuncios visualizados pueden variar según los siguientes factores:

- Ubicación geográfica
- Tipo de anuncio (Banner, intersticial, recompensados o nativos)
- CTR
- Tipo de dispositivo
- Categoría de la aplicación

Se realiza una estimación de los ingresos, suponiendo el uso únicamente de anuncios Intersticiales, que, en términos generales ofrecen unos 3€ cada mil visualizaciones.

- Se poseen unos 1000 usuarios recurrentes en la aplicación
- 300 de ellos la usan diariamente
- Los que la usan diariamente, visualizan 3 anuncios por dia
- Si visualizan 3 anuncios al día, por 300, serían 900 al día, por 30 días al mes, 27 mil, por 3 meses, 81 mil anuncios trimestrales fijos. 81 por 3 € da un total de 243€ fijos al trimestre en anuncios
- De los 300 que la usan diariamente, 20 pagan por suscripción trimestral

$$Beneficio\ Trimestral = 20\ x\ 35 + \ 243 = 943\ €$$

Por lo tanto, suponiendo una cantidad de 1000 usuarios con la aplicación descargada y 300 recurrentes (datos bastante reales), se obtendría un beneficio de 943€ por trimestre, lo que supone, 314,3€ al mes (sin incluir los anuncios que verían los otros 700 usuarios).

Viendo esta estimación, se concluye que es un plan de negocio muy bueno si previamente se realiza una campaña de publicidad para llegar al mayor número de usuarios posibles.

Para realizar campaña de publicidad existen numerosas posibilidades, entre las que encontramos:

- Facebook Ads, campaña muy buena para llegar al público específico, sabiendo que los usuarios de Facebook suelen ser de mayor edad
- Instagram Ads, campaña menos viable para conseguir usuarios premium, ya que los usuarios de Instagram tienen menor edad
- Google Ads, mejor campaña para conseguir referencias a la aplicación desde otras páginas o escalar posiciones en la Play store
- Televisión, muy viable para fases posteriores, cuando se poseen muchos usuarios y un buen presupuesto
- Físico en tiendas y por la calle, en muchas ocasiones gratuita pero con un mayor esfuerzo

De la misma forma, se debe realizar un buen SEO para las tiendas oficiales, consiguiendo escalar el mayor número de puestos posible con el mayor número de palabras clave. Las palabra clave utilizadas deben ser:

- Modelado 3D
- Diseño 3D
- Renderizado 3D
- Animación 3D
- Arte 3D
- Creación 3D
- Impresión 3D
- Software de modelado
- Gráficos 3D
- Herramienta 3D
- Modelos sólidos
- Dibujo 3D

En la siguiente ilustración se observa, según Google Trends, La búsqueda de palabras que contienen “3D”, viendo como se mantiene en el tiempo y no varía en gran medida.

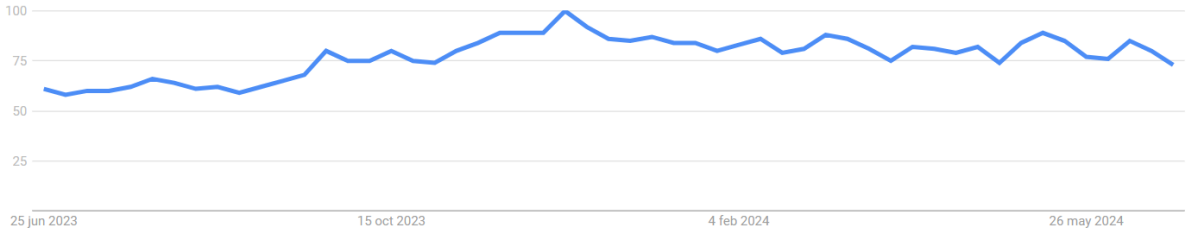


Ilustración 31 - Interés a lo largo del tiempo de palabras clave

5 CONCLUSIONES Y TRABAJOS FUTUROS

Tras realizar este TFG, se reflexiona sobre la importancia de la innovación en sectores actualmente en desarrollo, aprovechando oportunidades de negocio. De la misma forma hay que comprender la complejidad del TFG junto con todas las alternativas probadas, que son muchas viables, y una final aceptable para el ámbito que se abarca. Se han aplicado conocimientos de la mayoría de asignaturas que conforman el grado de ingeniería informática, utilizándolos para desarrollar el mismo y para mejorar otros existentes. Igualmente se entiende la fuerza que la inteligencia artificial está obteniendo en las tecnologías de la información, y, por último, se concluye en la increíble potencia que está llegando a obtener tecnologías que, hasta hace pocos años, se creían muy difíciles de alcanzar.

Como trabajo futuro, se debe mejorar los siguientes aspectos:

- Mejora de la infraestructura, aplicando uno o varios servidores en la nube con altas prestaciones
- Mejora de la “red social” dentro de la aplicación, permitiendo a usuarios interactuar entre ellos
- Permitir editar modelos para medir en tiempo real y así llegar a profesiones tan importantes como la medicina o la arquitectura

- Mejorar la calidad gráfica de los modelos, con más iteraciones en las redes neuronales, aplicando más capas ocultas o incluso creando una propia, dejando de lado Nerfstudio
- Pedir a los usuarios que registren mayor número de datos, como teléfono, lugar de residencia o DNI para poder asegurar con mayor certeza la seguridad de la aplicación y obtener más información del cliente
- Sincronización a tiempo real con drones. Actualmente la aplicación permite importar videos generados con drones, pero, no permite su sincronización directa con vuelos de drones, es un punto importante que se puede mejorar

6 APÉNDICES

6.1 Guía original del Trabajo Fin de Título

6.1.1 Descripción corta

Viajar es una aventura que nos transporta más allá de nuestro entorno cotidiano, permitiéndonos explorar otras culturas y paisajes. Cada viaje es una oportunidad de ampliar horizontes y compartir experiencias y nos brinda recuerdos imborrables.

Es muy frecuente que cada vez que el viajero descubre algo que le llama la atención, haga una fotografía para llevarse un recuerdo que le permita recrear ese momento de vuelta en casa. Sin embargo, en ocasiones las fotografías no capturan bien toda la esencia de eso que nos llamó la atención. ¿Y si fuese posible capturar imágenes mejoradas de las cosas que nos llaman la atención en los viajes, con la facilidad con la que les hacemos fotografías?

Este trabajo fin de grado consiste en desarrollar un prototipo de aplicación móvil que permita capturar fácilmente representaciones mas ricas que las fotografías clásicas de elementos reales que podamos encontrar en nuestros viajes, para poder observarlos posteriormente apreciando todo su esplendor.

6.1.2 Objetivos

- Reforzar los conocimientos adquiridos durante el grado sobre desarrollo de aplicaciones multiplataforma para dispositivos móviles.
- Aprender como realizar modelos en tres dimensiones de objetos, y cómo representarlos mediante técnicas de realidad aumentada.
- Estudiar las condiciones más habituales de las personas que viajan: entorno de uso del móvil, tiempo disponible para crear un recuerdo tridimensional, condiciones ambientales más habituales, capacidad de recargar los dispositivos, etc.
- Estudiar las necesidades más importantes de las personas que viajan a la hora de capturar recuerdos de los elementos de interés que encuentran durante su viaje.
- Elegir la mejor arquitectura para desarrollar el prototipo de la aplicación móvil.
- Desarrollar un prototipo de aplicación móvil para capturar recuerdos tridimensionales, teniendo en cuenta la experiencia de uso.

6.1.3 Metodología a desarrollar

1. Revisión de las tecnologías disponibles para realizar aplicaciones móviles multiplataforma.
2. Revisión de los framework disponibles para añadir realidad aumentada a una aplicación móvil.
3. Estudio de las condiciones de uso de los dispositivos móviles durante los viajes.
4. Estudio de las necesidades más importantes de las personas que viajan a la hora de capturar recuerdos de los elementos de interés.
5. Análisis del sistema, incluyendo requisitos funcionales y no funcionales.
6. Estimación de costes y planificación temporal.
7. Diseño de la arquitectura del sistema y de las interfaces de usuario.
8. Implementación de la solución.

9. Pruebas de software.
10. Redacción de la documentación final.

6.1.4 Documentos y formato de entrega

- Documentación generada en el proceso, de acuerdo a las buenas prácticas de Ingeniería del Software, en formato digital.
- Memoria del trabajo, incluyendo anexos y manuales de usuario e instalación, en formato digital.
- Código fuente y ejecutables.
- Documentos y archivos adicionales a los que se haga mención expresa en la memoria.
- Vídeo demostración del sistema.
- Informe favorable del tutor.
- Autorización de publicación, si procede.

6.2 Instalación y configuración del sistema

Para poner en marcha este sistema es necesario disponer de un ordenador personal con Windows o un servidor y de un teléfono smartphone con acceso a internet.

6.2.1 Puesta en marcha del servidor

El servidor debe disponer de las siguientes tecnologías:

- Python 3
- Anaconda
- Nerfstudio
- App con flask
- CUDA

En primer lugar se debe crear un entorno de trabajo con Anaconda, para ello es necesario ejecutar el comando *conda create nerfstudio,* se crea con el nombre “nerfstudio” (también es posible asignarle otro nombre). Seguidamente, se debe activar el entorno con *conda activate nerfstudio.* Una vez tengamos el entorno activado, se debe proceder a la instalación de CUDA.

Para instalar CUDA, hay que acceder a la página web oficial y en ella se encuentran los pasos a seguir para la instalación.

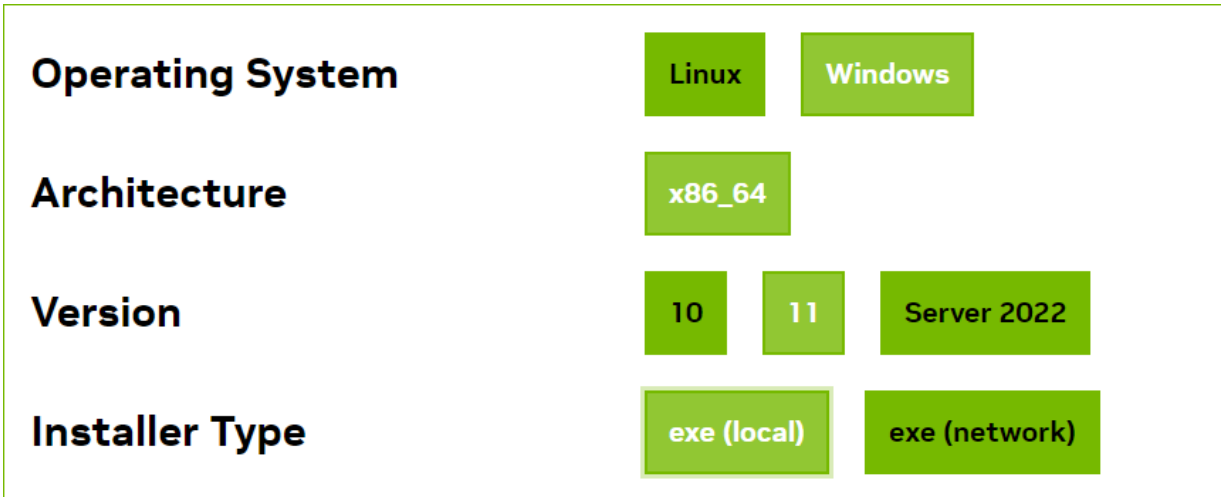


Ilustración 32 - Instalación CUDA

Una vez se descargue e instale el ejecutable de CUDA, es necesario instalar nerfstudio (hay que tener cuidado con las versiones de ambos para que no haya conflictos). Para instalar Nerfstudio es necesario acceder a su página web oficial y seguir los pasos de instalación.

Con nerfstudio instalado y configurado, ya se puede poner en marcha la aplicación Flask para el servidor. Es recomendable crear una carpeta para tener en su interior todo lo relativo al servidor y a este sistema con las siguientes subcarpetas y archivos:

- Images, contendrá todos los archivos que el servidor recibirá
- Outputs, se crearán aquí los modelos
- App

  Se debe integrar un archivo Python con el código para la lógica del servidor.

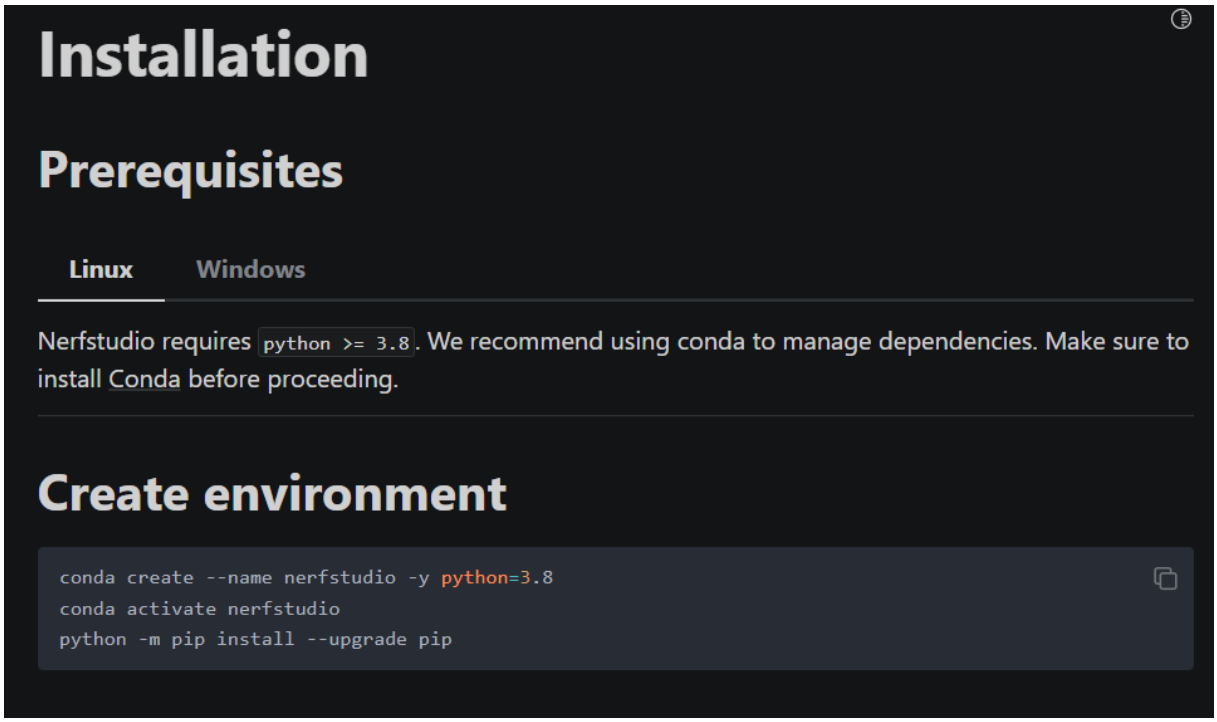


Ilustración 33 - Web oficial Nerfstudio

6.2.2 Instalación de la aplicación

La aplicación se puede instalar directamente en el teléfono, activando la opción de “Orígenes desconocidos” para permitir su instalación directa. En el caso de un futuro lanzamiento a las tiendas oficiales, no sería necesario esto ya que esa seguridad está garantizada por la propia tienda. Es importante cambiar la dirección IP de la aplicación en su código fuente, ya que originalmente contiene la del servidor de prueba.

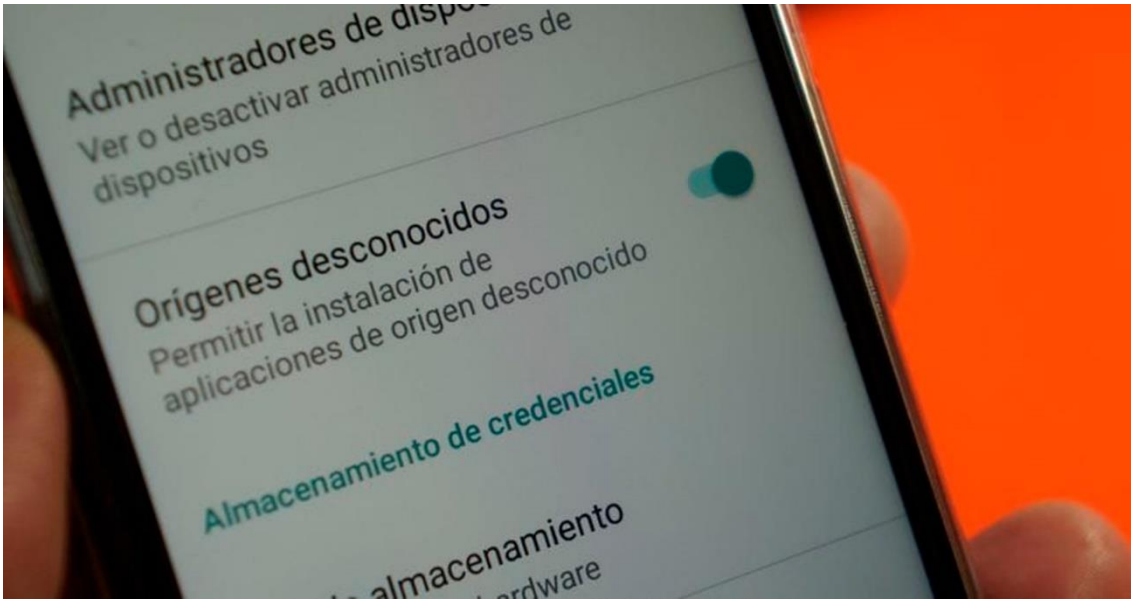


Ilustración 34 - Activar orígenes desconocidos

6.3 Manuales de usuario

En esta sección se explica como utilizar la aplicación y sus funcionalidades.

6.3.1 Inicio de sesión

Para iniciar sesión es necesario poseer una cuenta en la aplicación, se debe introducir el correo con el que se creó la cuenta y la contraseña elegida.

Ilustración 35 - Manual de usuario - Inicio de sesión

Tras introducir los credenciales, se debe pulsar en el botón “Iniciar Sesión” para iniciar sesión en la aplicación.

6.3.2 Crear cuenta

Para crear una cuenta, en primer lugar se debe pulsar sobre el botón crear cuenta.

Ilustración 36 - Manual de usuario - Crear cuenta botón

Se abrirá una pantalla para introducir credenciales como el correo y la contraseña, tras introducirlos, se debe pulsar sobre “Crear Cuenta”.

Ilustración 37 - Manual de usuario - Crear cuenta

Una vez pulsado e introducido con éxito los credenciales, se habrá creado una cuenta y enviado un correo de verificación de la misma.

6.3.3 Verificar cuenta

Para verificar la cuenta creada basta con entrar en el correo con el que se ha creado la cuenta y pulsar sobre el enlace de verificación.

Ilustración 38 - Manual de usuario - Verificar email

6.3.4 Crear modelo con cámara

Crear un modelo nuevo es muy sencillo, primero debe estar la cuenta verificada para poder acceder a esta funcionalidad. Hay que pulsar sobre el botón crear modelo y elegir la opción “Cámara”.

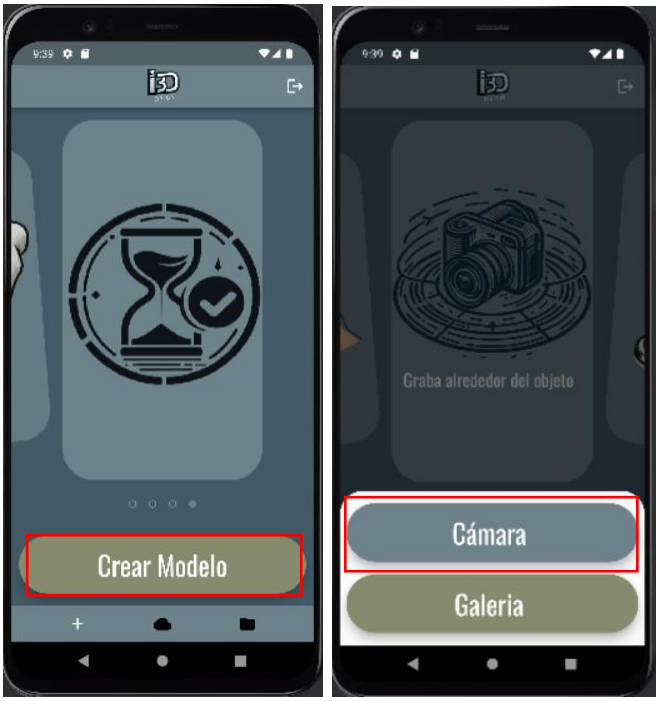


Ilustración 39 - Manual de usuario - Crear modelo cámara

El siguiente paso es grabar alrededor del objeto a modelar (cuanta más exactitud mejor). Posteriormente hay que asignarle un nombre al nuevo modelo y pulsar en “Aceptar”.

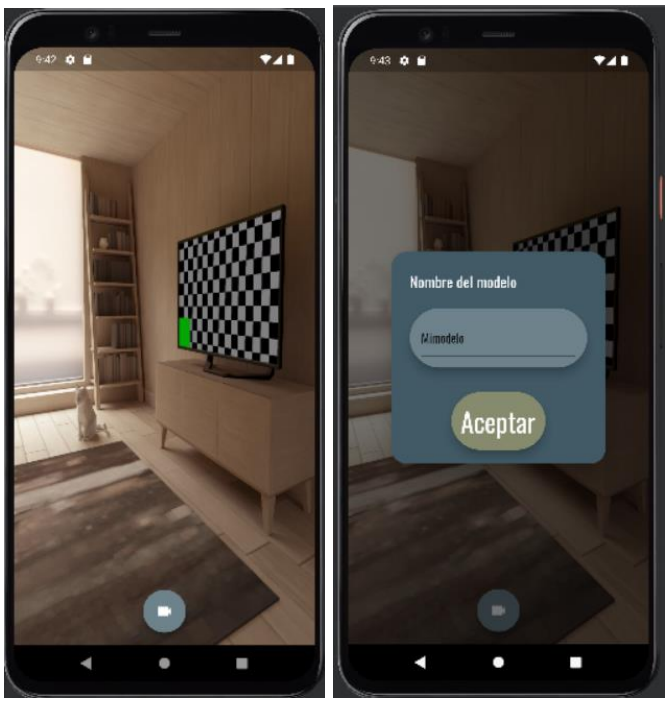


Ilustración 40 - Manual de usuario - Botón aceptar crear modelo

Como último paso, se redirigirá a una pantalla con un botón en el centro que pone “Comenzar construcción”. Se debe pulsar sobre dicho botón y esperar la creación del modelo.

Ilustración 41 - Manual de usuario - Botón comenzar construcción

6.3.5 Visualizar modelo local

Para visualizar un modelo localmente se debe acceder al icono de la carpeta situado en la parte inferior derecha y elegir el modelo a visualizar (necesario haber creado previamente un modelo).

Ilustración 42 - Manual de usuario - Icono carpeta

A continuación, seleccionar el modelo deseado.

Ilustración 43 - Manual de usuario - Seleccionar modelo

Y, con ello, se puede visualizar el modelo.

Ilustración 44 - Manual de usuario – Visualizado

6.3.6 Subir modelo

Subir un modelo es muy sencillo, basta con presionar sobre el icono de la nube cuando se esté visualizando uno y, seguidamente, pulsar sobre “Sí”.

Ilustración 45 - Manual de usuario - Subir modelo

6.3.7 Eliminar modelo local

Para eliminar un modelo, en la pantalla “Mis modelos”, hay que deslizar el modelo a eliminar de izquierda a derecha (aparecerá un icono de una papelera indicando que está borrando dicho modelo).

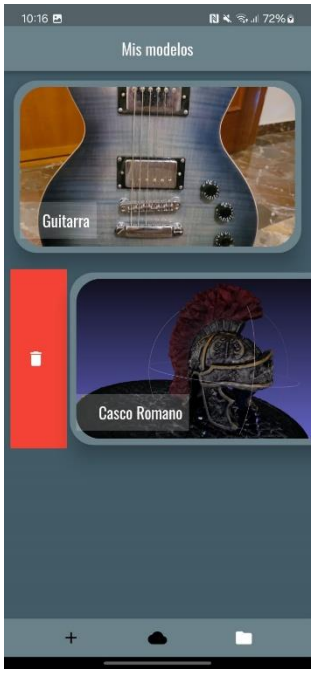


Ilustración 46 - Manual de usuario - Eliminar modelo

6.3.8 Descargar modelo

También existe la posibilidad de descargar un modelo de la nube, para ello, navegar al icono de la nube y presionar sobre el modelo deseado.

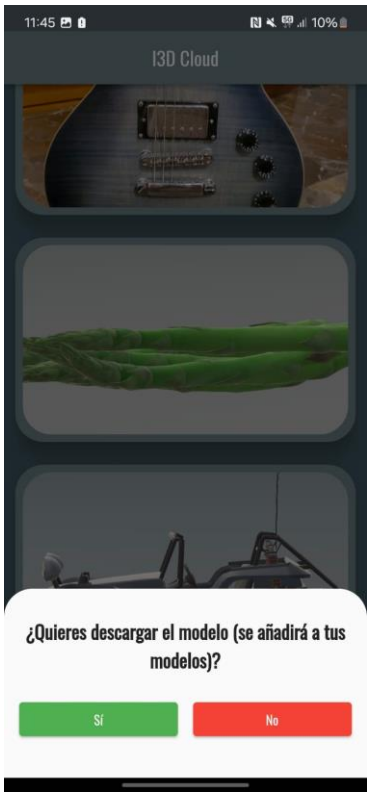


Ilustración 47 – Manual de usuario - Descargar modelo

Para visualizar el modelo descargado, hay que navegar de nuevo a la pantalla de “Mis modelos” y seleccionarlo.

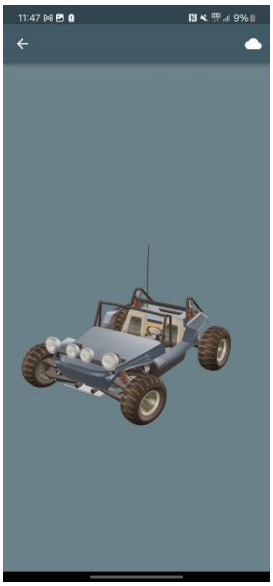


Ilustración 48 - Manual de usuario - Ver modelo descargado

7 DEFINICIONES Y ABREVIATURAS

Blender: Software de diseño de modelos 3D

IOs: Sistema operativo desarrollado por Apple

LIDAR: Light Detection and Ranging

Deep Learning: Aprendizaje Profundo, redes neuronales

SIFT: Scale-invariant feature transform

BFMatcher: detector de coincidencias

AliceVision: Framework de fotogrametría

API: Application Programming Interface

NGINX: Servidor HTTP y de proxy inverso, servidor proxy de correo y servidor TCP/UDP

Ingeniero QA: Quality Assurance

UI: User Interface

HTTP: Hyper Text Transfer Protocol

SQL: Structured Query Language

Figma: Software para MockUps

IA: Inteligencia Artificial

Dall-e: Inteligencia Artificial para la generación de imágenes

GitHub: Plataforma de desarrollo colaborativo

CUDA: Compute Unified Device Architecture

NoSQL: Not Only SQL

8 BIBLIOGRAFÍA

[1] Learn Google Flutter Fast, 65 Examples Apps

[2] Flutter Aprentice, First Edition, Raywenderlich Tutorial Team

[3] Flutter Aprentice, Second Edition, Raywenderlich Tutorial Team

[4] Web Oficial de Documentación de Nerfstudio, referencia: https://docs.nerf.studio/

[5] Web Oficial de Polycam, referencia: https://poly.cam/

[6] Transparencias de la asignatura “Informática Gráfica y Visualización”, Juan José Jiménez Delgado

[7] Transparencias de la asignatura “Procesamiento de la Información Visual”, Manuel Lucena López

[8] Convenio Colectivo Estatal de la Industria y la Tecnología

[9] Ministerio de empleo y Seguridad Social, Tablas salariales 2023

[10] R. Pressman, Ingeniería del Software, McGraw-Hill, 2010.