



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO
GRADO EN INGENIERIA INFORMATICA

EditAR (working tittle)

Editor de escenas de Realidad Aumentada

Autor

Pablo Millán Cubero

Director

Francisco Javier Melero Rus



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, Julio de 2023

EditAR

Editor 3D web para escenas de Realidad Aumentada

Pablo Millán Cubero

Palabras clave: *software libre, informática gráfica, Typescript, React, Firebase, NodeJS, Express, Android Studio, Kotlin, Sceneview, GLB*

Resumen

Las tecnologías de Realidad Aumentada nos permiten combinar elementos virtuales y reales en tiempo real pudiendo así crear experiencias aplicables a muchos campos; filtros con maquillaje para fotos en redes sociales, previsualizar cómo quedaría un mueble en tu propio salón, dibujar en un partido de fútbol la trayectoria que recorrió la pelota en una repetición, o incluso en videojuegos como Pokemon GO.

Con el avance de los teléfonos móviles de todas las gamas en cuanto a cámaras y capacidad de procesamiento en la última década estas experiencias han pasado a estar al alcance de todo el mundo que posea un dispositivo básico con resultados muy vistosos. Sin embargo en la mayoría de experiencias de Realidad Aumentada el usuario ocupa el rol de consumidor pasivo del contenido que generan los desarrolladores.

Se propone entonces desarrollar una aplicación web en el que el usuario pueda cargar múltiples modelos 3D con textura y animaciones, aplicarles a estos transformaciones como translaciones, rotaciones y escalado, reproducir animaciones y cargar pistas de audio. Todo ello desde un sencillo e intuitivo editor que puede usar cualquiera en el que no haga falta tener conocimientos previos. La escena 3D que se cree podrá posteriormente descargarse o guardarse en un servidor web asociado a una cuenta de usuario. El usuario podrá además, desde una aplicación Android cargar sus escenas creadas y reproducirlas, pudiendo visualizarlas en su entorno a través de la cámara del dispositivo móvil.

EditAR **Web editor for Augmented Reality scenes**

Pablo Millán Cubero

Keywords: *software libre, informática gráfica, Typescript, React, Firebase, NodeJS, Express, Android Studio, Kotlin, Sceneview, GLB*

Abstract

Augmented Reality technologies let us merge virtual elements with camera footage in real time, letting us create experiences for many fields; makeup filters for social network photos, preview new furniture for your livingroom, draw the trajectory of the ball in a football match replay, or even in videogames like Pokemon GO.

With the improvement of smartphone's cameras and processing power in the last decade, this experiences are know within everybody's reach if you have a basic device. However, the majority of Augmented Reality experiences are based in the pasive consumption of content pregenerated by developers.

The proposed web application lets the user load multiple 3D models with textures and animations, apply transformations like translations, rotations and scales, and play audio tracks. All this in an easy and intuitive editor that anyone can use without previous knowledge. The resultant 3D scene can be either downloaded or uploaded to a web server asociated to an user account. Additionaly, the user can load the created scenes within an Android app and play them with their device's camera, placing it in their environment.

D. Tutora/e(s), Profesor(a) del ...

Informo:

Que el presente trabajo, titulado **Chief**, ha sido realizado bajo mi supervisión por **Estudiante**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2018.

El/la director(a)/es:

(nombre completo tutor/a/es)

Agradecimientos

A mis padres por permitirme estudiar esta carrera en Granada y apoyarme.

A mis abuelos, en especial a mi abuela Elena, que tanto presume de "su nieto el ingeniero".

A mis tíos "los franceses".

A mis amigos del Krustáceo Krujiente, Juan, Antonio, y mis compañeros de trinchera Pepe, Finn, Moisés y Pablo.

Índice general

1. Introducción y fundamentos	17
1.1. Descripción general y motivación	17
1.2. Fundamentos	18
1.2.1. Realidad Aumentada	18
1.2.2. Visión por computador	21
1.2.3. ARCore	22
1.2.4. SceneView	23
1.2.5. Android y Android Studio	23
1.2.6. Typescript	24
1.2.7. Archivo JSON	24
1.2.8. Three.js	25
1.2.9. React.js	25
1.2.10. Node.js	25
1.2.11. Express	25
1.2.12. Firebase	25
1.2.13. Archivo GLB	26
2. Descripción del problema	27
2.1. Problema a resolver	27
2.2. Objetivos	28
2.3. Estado del arte	29
2.3.1. Onirix Studio	29

2.3.2.	Blippbuilder	31
2.3.3.	Pictarize	32
2.4.	Crítica	33
2.5.	Propuesta	35
3.	Estado de la Realidad Aumentada	37
4.	Planificación	39
4.1.	Metodología utilizada	39
4.2.	Historias de usuario	41
4.2.1.	RA-01: Entorno 3D	43
4.2.2.	RA-02.1: Escenas de superficie	43
4.2.3.	RA-02.2: Escenas de imágenes aumentadas	43
4.2.4.	RA-02.3: Escenas geoespaciales	44
4.2.5.	RA-03: Varias imágenes activadoras	44
4.2.6.	RA-04: Animaciones	44
4.2.7.	RA-05: Pistas de audio	44
4.2.8.	RA-06: Carga de modelos	45
4.2.9.	RA-07: Transformaciones	45
4.2.10.	RA-08: Selección múltiple	45
4.2.11.	RA-09: Figuras básicas	45
4.2.12.	RA-09.1: Texturas	45
4.2.13.	RA-10: Inicio de sesión	46
4.2.14.	RA-10.1: Cambio de credenciales	46
4.2.15.	RA-10.2: Guardado de escenas	46
4.2.16.	RA-11.1: Reproducción de imágenes aumentadas	46
4.2.17.	RA-11.2: Reproducción de escenas por superficies	47
4.2.18.	RA-11.3: Reproducción de escenas geoespaciales	47
4.2.19.	RA-12: Inicio de sesión en la app móvil	47
4.2.20.	RA-12.1: Listado de escenas	47
4.2.21.	RA-12.1: Filtrado de escenas	47
4.2.22.	RA-13: Exportación de escenas	48
4.3.	Sprint 0	48

4.4. Sprint 1	52
4.5. Sprint 2	54
4.6. Sprint 3	56
4.7. Sprint 4	57
4.8. Sprint 5	60
5. Implementación	63
6. Conclusiones y trabajos futuros	65

Índice de figuras

1.1. Ejemplo RA	19
1.2. Ejemplo ground	19
1.3. Ejemplo ground	20
1.4. Ejemplo ground	21
2.1. Aplicación Onirix Studio	30
2.2. Aplicación Onirix Studio	32
2.3. Aplicación Onirix Studio	33
2.4. Gizmo en Autodesk AutoCAD	34
4.1. Planificación de HU por sprints	49
4.2. Tabla de sprint	50
4.3. Burndown Sprint 1	54
4.4. Burndown Sprint 2	55
4.5. Burndown Sprint 3	57
4.6. Burndown Sprint 4	59
4.7. Burndown Sprint 5	61

Índice de tablas

Capítulo 1

Introducción y fundamentos

En este capítulo se dará una primera introducción a la *Realidad Aumentada* y a la idea del proyecto, además de describir una serie de conceptos y tecnologías fundamentales para entender el trabajo y seguir con la lectura del resto del documento.

1.1. Descripción general y motivación

El término de Realidad Aumentada comprende al conjunto de técnicas y tecnologías que permiten la superposición e interacción de elementos virtuales sobre la realidad física a través de un dispositivo electrónico. Esta tecnología se ha aplicado a una gran variedad de ámbitos, tales como en la creación de filtros faciales para fotos en redes sociales como Instagram, dibujar la trayectoria que ha seguido el balón en la repetición de una jugada de fútbol, muestra de distintos gráficos y maquetas en los telediarios, previsualización de mobiliario en una habitación e incluso videojuegos como Pokémon GO.

El avance en las cámaras y procesadores de los teléfonos móviles en la última década ha facilitado que casi cualquier persona tenga en el bolsillo un dispositivo capaz de reproducir experiencias muy vistosas de Realidad Aumentada. Sin embargo, en la mayoría de casos se trata de aplicaciones en las que el usuario consume pasivamente contenido pre generado por los desarrolladores, sin la capacidad de crear ellos mismos estas experiencias.

Se propone entonces desarrollar en primer lugar una aplicación web que permita al usuario la composición de textitescenas 3D para uso en Realidad Aumentada. Cuando hablamos de escena nos referimos una composición de uno o más modelos 3D con una disposición concreta. Por ejemplo podríamos colocar el modelo de un tobogán junto al de un columpio y un balancín, formando el

conjunto la escena de un parque. La aplicación consistirá principalmente de un editor sencillo e intuitivo que permita cargar modelos 3D con texturas y aplicarles transformaciones para colocarlos en una escena. Adicionalmente, se podrá añadir a la escena una pista de audio y reproducir animaciones de los modelos. Se podrá guardar la escena en un servidor web en una colección de escenas generadas asociadas a una cuenta de usuario, desde la cual se podrán modificar o eliminar.

Adicionalmente, el usuario podrá iniciar sesión en una app móvil Android para acceder a su lista de escenas creadas y podrá reproducir cualquiera de ellas en Realidad Aumentada con su dispositivo móvil.

Las principales motivaciones para elegir este como mi trabajo de fin de grado fue mi alto grado de interés por los gráficos 3D, mis conocimientos previos en tecnologías que acabaría usando como three.js o Android Studio, y que es un proyecto relativamente multidisciplinar, requiriendo del desarrollo de una aplicación web, una aplicación Android y un servidor web que los conecte, montando así un pequeño ecosistema y con suerte dándome una intuición de lo que puede ser realizar un proyecto parecido en un entorno laboral real.

1.2. Fundamentos

A continuación se van a describir una serie de conceptos y tecnologías esenciales para la correcta comprensión del proyecto.

1.2.1. Realidad Aumentada

La *Realidad Aumentada* (*Augmented Reality* en inglés, *RA* de aquí en adelante) hace referencia al conjunto de técnicas y tecnologías que permiten crear una experiencia en la que se combina el mundo real con contenido generado por computadores. Aunque el uso más extendido es únicamente visual y auditivo, puede apelar a más sentidos como el háptico, somatosensorial y olfativo. Se puede decir que un sistema *RA* debe incorporar tres elementos básicos: una combinación del mundo real y el virtual, interacción en tiempo real y un correcto registro de la posición y movimientos de objetos tanto reales como virtuales.

Existen otros términos acuñados que son a grandes rasgos sinónimos de la *RA* como *Mixed Reality*. No debe confundirse con la *Realidad Virtual*, otro ámbito en el que el objetivo es sumergir al usuario en un mundo virtual enajenándolo de sus sentidos con el mundo real, usualmente con unas gafas envolventes diseñadas específicamente para realidad virtual. Todas estas disciplinas están recogidas bajo el término paraguas de la *Extended Reality (XR)*.

A continuación se van a describir los tipos de escenas que podemos encontrarnos en el contexto de la Realidad Aumentada.



Figura 1.1: Ejemplo de Realidad Aumentada en el videojuego Pokémon Go

- **Posicionamiento en superficies:** Es quizás el ejemplo más sencillo tanto conceptualmente como técnicamente. El dispositivo intentará analizar sus alrededores a través de la cámara e intentará detectar superficies planas como podría ser una el suelo, una mesa o una pared. Colocará objetos en puntos concretos de estas superficies, y al mover el dispositivo esos objetos deberán visualizarse como si estuvieran colgados o colocados en dichas instancias del mundo real. Existe todavía a día de hoy una limitación importante a la hora de realizar estas escenas; si la superficie es demasiado lisa y uniforme, el dispositivo la reconocerá con gran probabilidad como un color plano, y no tendrá la capacidad de identificarla como una superficie, haciendo que no pueda llegar a colocar nada.

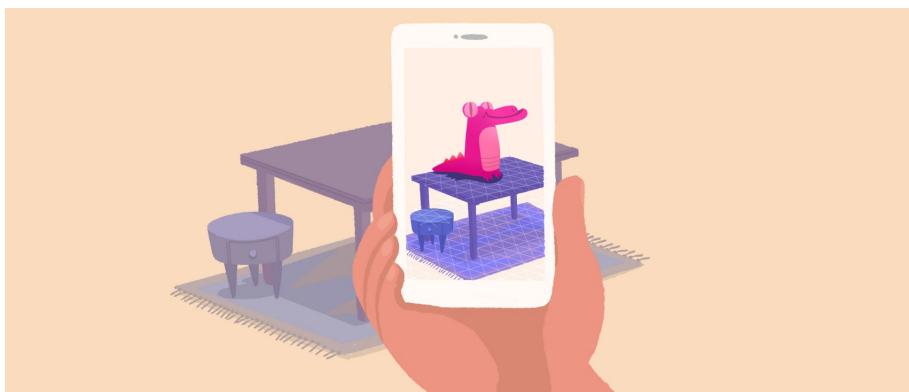


Figura 1.2: Ejemplo de posicionamiento en superficies de la documentación de ARCore [6]

- **Imágenes aumentadas:** En este tipo de escena, cada modelo o conjunto de modelos tiene asociada una imagen bidimensional a la que denominamos **marcador** o **activador**. Cuando se inicie, el dispositivo tratará de reconocer esta imagen en el mundo real. Una vez identificada se reproducirán sobre la imagen (tratándola como una superficie) o en una posición relativa a la imagen, los modelos pertinentes que se hayan definido. Es posible definir distintos objetos asociados a distintos marcadores y que solo se reproduzcan los elementos pertinentes.



Figura 1.3: Ejemplo de escena de imágenes aumentadas de la web de *Pictarize Studio* [1]

- **Escenas geoespaciales:** Son escenas en las que los objetos no se posicionan en una superficie, si no en unas coordenadas GPS. El usuario solo podrá ver la escena si se encuentra a una latitud, longitud y altura cercanos a aquella que se definieran para la escena. Para ello logicamente es esencial disponer de conexión a internet y que el dispositivo tenga acceso a las coordenadas actuales del usuario durante la ejecución. Las localización GPS de los dispositivos móviles no suele ser del todo precisa, así que usando solo estas es posible que si queremos, por ejemplo, colocar un modelo 3D que represente cómo era la *Alhambra de Granada* cuando se construyó justo en la puerta del edificio, es posible que algunos usuarios la vean algunos metros fuera de lugar, o incluso flotando en el aire. Para poder tener algo más de precisión, algunas APIs geoespaciales como la de *ARCore* [6] permiten extraer información de *Street View*, el visor en primera persona de *Google Maps* para reconocer a través de puntos clave,

fachadas de edificios y calles que hayan sido captadas previamente por este servicio, resultando así en una mejor aproximación a la verdadera posición del usuario y por tanto un posicionamiento más preciso de las escenas.

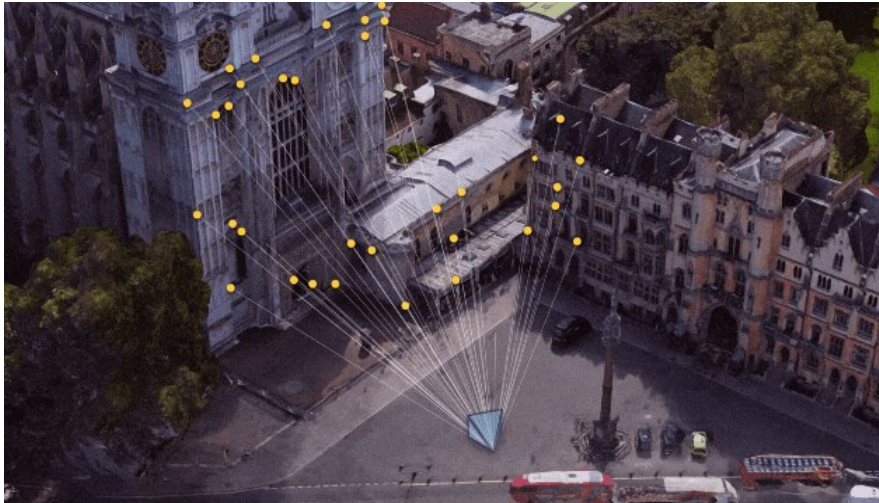


Figura 1.4: Ejemplo de la aproximación por puntos clave para escenas geoespaciales de la documentación de ARCore [6]

1.2.2. Visión por computador

Javier González Jiménez define en su libro *Visión por computador* [9] este campo con la siguiente cita:

*"La visión por computador, también denominada visión artificial, puede definirse por el proceso de extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador. Desde un punto de vista más ingenieril, un sistema de visión por computador es un sistema autónomo que realiza algunas de las tareas que el sistema de visión humano [...] realiza. La información o tareas que este sistema de visión puede llegar a extraer o realizar puede ir desde la simple detección de objetos sencillos en una imagen hasta la **interpretación tridimensional de complicadas escenas.**"*

"[...] la información visual consiste en energía luminosa procedente del entorno. Para utilizar esa información es preciso transformarla a un formato susceptible de ser procesado [...] esta misma tarea se realiza con una cámara de vídeo (o algún otro dispositivo similar), que convierte la energía luminosa en corriente eléctrica, que puede ser entonces muestreada y digitalizada para su procesamiento en un computador."

En nuestro caso nos interesa especialmente la sentencia destacada de la cita. Esta disciplina nos permite extraer una idea de la estructura física de un escenario,

permitiéndonos detectar elementos como superficies o esquinas, indispensables si queremos por ejemplo, posicionar un objeto 3D encima de una mesa, y que el objeto parezca que siga apoyado en la mesa si movemos la cámara de lugar, cambiamos el ángulo, o nos alejamos.

Por fortuna la obtención de información a través de imágenes será un proceso completamente transparente para nosotros gracias a las distintas herramientas y tecnologías que ofrecen una API para hacer llamada de estas funciones sin que tengamos que preocuparnos por los detalles técnicos como veremos a continuación.

1.2.3. ARCore

ARCore [6] es un *SDK* (*software development kit*) de Realidad Aumentada desarrollado por Google. Ofrece una API con la que se pueden desarrollar experiencias AR en Android, iOS, Unity y Web renderizando modelos 3D con *OpenGL*. Para ello ARCore realiza una serie de tareas a bajo nivel relativas a la visión por computador y por las cuales el usuario no tiene que preocuparse, solo hacer las llamadas necesarias a su API. Estas funciones se detallan a continuación.

- **Seguimiento de movimiento:** Se emplea un proceso llamado *localización y asignación simultáneos*, o ANSM, para que el dispositivo obtenga un contexto sobre el mundo físico que lo rodea. Para ello ARCore detecta características visualmente distintas en las imágenes capturadas (también llamadas **puntos de atributo**) y los usa para calcular el cambio de ubicación del dispositivo. La información visual se combina con la IMU¹ del dispositivo para calcular la pose, es decir, la posición y orientación de la cámara respecto al mundo a lo largo del tiempo.
- **Comprensión ambiental:** Búsqueda de clústeres de puntos de atributos que aparentan pertenecer a superficies horizontales o verticales como parecen o mesas, y las pone a disposición del usuario como *planos* que pueden emplearse para posicionar objetos.
- **Comprensión de profundidad:** ARCore detecta y almacena *mapas de profundidad* con información sobre la distancia entre dos puntos cualquiera pertenecientes a una superficie. Esto puede emplearse para simular interacciones físicas realistas entre objetos virtuales y reales o lograr el efecto de que un objeto virtual aparezca tapado o detrás de uno real.
- **Estimación de luz:** Detecta información sobre la luz de su entorno, su intensidad y color. Con esto se pueden lograr efectos como que un objeto

¹Un IMU (Unidad de Medida Inercial) es un dispositivo capaz de estimar y reportar información acerca de la velocidad y orientación del mismo a través de acelerómetros y giroscopios.

virtual a la sombra se represente más oscuro que uno expuesto a una fuente de luz.

- **Estimación de luz:** Detecta información sobre la luz de su entorno, su intensidad y color. Con esto se pueden lograr efectos como que un objeto virtual a la sombra se represente más oscuro que uno expuesto a una fuente de luz.
- **Interacción de usuario:** A través de raycasts², ARCore puede obtener un *punto de atributo* o modelo posicionado en la escena a partir de un gesto de usuario como tocar la pantalla táctil del dispositivo, lo que permite interacciones como colocar o mover objetos 3D.
- **Puntos orientados:** Los puntos orientados permiten posicionar objetos 3D en superficies que no son superficies planas. ARCore analiza los puntos de atributo vecinos al seleccionado para posicionar el objeto y realizará una estimación de la inclinación, devolviendo un ángulo que servirá para aplicar una pose al objeto para que se vea lo más natural posible.
- **Imágenes aumentadas:** ARCore puede detectar a través de la cámara imágenes 2D definidas previamente y asociar a cada una uno o varios modelos 3D que posicionar sobre estas en el caso de que se enfoquen. Los modelos mantendrán su posición de forma consistente si la cámara se mueve o rota.

1.2.4. SceneView

Sceneform es un *SDK* desarrollado por *Google* que permite importar y visualizar modelos 3D en distintos formatos para renderizar escenas 3D para aplicaciones de *ARCore* o realidad virtual sin necesidad de programar código *OpenGL* de bajo nivel. Cuenta con un *grafo de escena* a través del cual podemos definir la estructura de los modelos de la misma a través de un árbol de nodos, y un motor de físicas de *Filamente*³.

Lamentablemente, *Sceneform* fue abandonado por *Google*, pero el proyecto ha renacido bajo el nombre de **Sceneview** [8], desligándose de *Google*, con código abierto y mantenido por su comunidad.

1.2.5. Android y Android Studio

Android es un sistema operativo desarrollado por Google, de código abierto y basado en el núcleo de *Linux*. En un principio se diseñó con dispositivos táctiles

²text raycast

³Filament: Motor de físicas para escenas 3D en tiempo real desarrollado por Google.

en mente como teléfonos inteligentes y *tablets*. Es el que actualmente tiene la mayor cuota de mercado en dispositivos de este tipo, contando en 2022 con el 72,2%⁴. También se emplea en otro tipo de sistemas como televisiones y relojes inteligentes o incluso automóviles.

Android Studio es un entorno de desarrollo integrado o *IDE*, es decir, una aplicación informática que proporciona servicios y recursos para el desarrollador de software como un editor de código, depurador, compilador e interfaz gráfica. En este caso para el desarrollo de aplicaciones en dispositivos Android. Este tipo de herramientas es de gran interés para encarar un desarrollo, ya que aglutina todas las herramientas que podemos necesitar como programadores en un solo entorno.

Kotlin es un lenguaje de programación multiplataforma, estáticamente tipado, con inferencia de tipos y de alto nivel. Está diseñado para ser totalmente interoperable con *Java*, pero usando una sintaxis más concisa. Es además, el lenguaje preferido por Google para el desarrollo de aplicaciones Android y el que se recomienda en toda la documentación.

1.2.6. Typescript

JavaScript (JS) es un lenguaje de programación interpretado y compilado *just-in-time*, aunque es más conocido como un lenguaje de *scripting*, es decir, un lenguaje que nos permite incrustar código dentro de páginas web.

Por otro lado, *TypeScript (TS)* es un lenguaje construido por encima de JavaScript que añade sintaxis adicional que lo hace **fuertemente tipado**. Es decir, siempre que declaremos una variable debemos especificar su tipo. Si más adelante en el código se intenta asignar un valor no correspondiente a ese tipo, el editor podrá detectarlo y avisar, pudiendo así identificar posibles errores antes siquiera de la ejecución del programa, cosa que con *JavaScript* no es posible.

1.2.7. Archivo JSON

Los archivos JSON (o *JavaScript Object Notation*) es un formato de texto sencillo para el intercambio de datos. Toma como referencia la notación de objetos de *JS*, pero a lo largo de los años y debido a su simplicidad, se ha convertido en la opción por defecto para intercambio de información entre servicios web, en contraposición a su principal alternativa, *XML*.

⁴<https://root-nation.com/es/noticias-es/es-cuota-mercado-android-ios-estadisticas-publicadas->

1.2.8. Three.js

Three.js [?] es una librerías *JavaScript* usada para crear y renderizar gráficos 3D animados en páginas web para que puedan se reproducidas en un navegador. Para ello hace uso de *WebGL*, otra librería *JavaScript* para la renderización de gráficos pero a más bajo nivel. *Three.js* se abstrae de conceptos técnicos y de bajo nivel para que el desarrollador no tenga que preocuparse de todo eso y agilice su desarrollo. Además es una librería de código abierto mantenida por su comunidad.

1.2.9. React.js

React [?] es una librería de *JavaScript* para construir interfaces de usuario web de código abierto y mantenido por su comunidad. Opcionalmente, utiliza ficheros de extensión *.jsx* (*JavaScript Syntax Extension*) para facilitar el desarrollo. Estos son, como su nombre indica, una extensión sintáctica de *JS* que permite insertar código HTML, facilitando así la construcción de interfaces de usuario.

1.2.10. Node.js

Node [5] es un entorno de ejecución de *JavaScript* orientado a eventos asíncronos diseñado para crear aplicaciones de red rápidas capaces de manipular grandes cantidades de conexiones concurrentes con una alta escalabilidad para los proyectos.

1.2.11. Express

Express [3] es un “*Web application framework minimalista y flexible para Node.js que provee una serie de herramientas para aplicaciones web y móviles.*” como lo definen ellos mismos. Cuenta con una gran variedad de utilidades para llamadas *HTTP* y *middleware*. Cuando hablamos de *middleware* nos referimos a un software a través del cual distintas aplicaciones se comunican entre sí por internet.

1.2.12. Firebase

Firebase [7] es un BAAS⁵ desarrollado por *Google* que pone a disposición de los desarrolladores una serie de herramientas y servicios para facilitar la creación

⁵Backend-as-a-service: aplicaciones puestas al servicio de desarrolladores para que estos hagan usos de sus servicios y se centren en desarrollar el frontend de sus apps, siendo innecesario así que creen ni mantengan sus propios backends.

de aplicaciones. Algunas de sus servicios más importantes son:

- **Autenticación:** Soporta autenticación de usuarios usando contraseñas con correo electrónico, número de teléfono, o cuentas de *Google*, *Facebook* y *Twitter* entre otros. Ofrece también encriptación por defecto en la base de datos para contraseñas.
- **Base de datos en tiempo real:** *Firebase* tiene también una base de datos **no relacional** que se actualiza a tiempo real y se mantiene en línea incluso cuando la aplicación no se está ejecutando.
- **Hosting:** Permite *hostear* aplicaciones web de forma fácil y rápida, pudiendo almacenar contenido en una memoria caché y accesible desde cualquier parte del mundo.

1.2.13. Archivo GLB

GLB (*.glb*) es un formato de archivo binario estandarizado para representar datos de objetos 3D como modelos, escenas, iluminación, materiales, jerarquías y animaciones. Sus siglas significan *GL Transmission Format Binary File*. Fue introducido en 2015 como una versión binaria de los ficheros **GLTF** (*.gltf*), un formato basado en la notación *JSON* muy usado en aplicaciones web y móvil debido a su ligero peso.

Capítulo 2

Descripción del problema

En este capítulo se detallará el problema que se pretende resolver con este proyecto, además de establecer los objetivos que se requieren cumplir. Se realizará una crítica al estado del arte y para finalizar se enunciará una propuesta de producto.

2.1. Problema a resolver

Como se ha comentado en apartados anteriores, existen en el mercado multitud de experiencias de Realidad Aumentada al alcance de los usuarios, aunque en la gran mayoría de ellas los usuarios juegan un rol pasivo. Esto último no hace referencia a que no puedan interactuar con el contenido, (ya hemos visto que la interacción en tiempo real es uno de los rasgos definitorios de la *RA*) sino que consumen contenidos y experiencias que han sido creadas por los desarrolladores de las distintas apps. Suelen ser sistemas cerrados donde el usuario no tendría opción de, por ejemplo, cargar un modelo 3D de su dispositivo para visualizarlo en su salón.

Por otro lado, las herramientas de modelado y animación de objetos 3D como *Blender*¹ o *3ds Max*² son herramientas profesionales que requieren de una formación, conocimientos previos y curva de aprendizaje para poder manejarlos con soltura. Incluso si las aplicaciones pudieran efectivamente cargar modelos en las aplicaciones ya mencionadas, estarían limitados a elementos sueltos que pudieran encontrar en internet. Esto es algo que limita mucho tus opciones creativas. Volviendo al ejemplo de la escena del parque que se puso en el capítulo 1, si se pretende recrear hacer la escena de un parque, quizás se encuentre un

¹link blender

²link 3ds max

modelo 3D para ello. Pero, ¿y si se quiere tenga concretamente dos toboganes y un columpio? Es muy poco probable que alguien haya modelado y publicado una escena con exactamente los mismos requisitos. Pero contando con que se puedan encontrar los elementos por separado (un modelo de un tobogán, de un balancín, de un columpio...) lo cual es más realista, y alguna forma de componerlos en una sola escena, se podría construir un parque con una composición cualquiera.

Teniendo todo esto en cuenta se detectan dos necesidades para el usuario. Por un lado, una suerte de editor sencillo, simplificado e intuitivo en el que un usuario sin conocimiento o experiencia previa pueda cargar o crear modelos 3D con los que pueda construir una escena con total libertad creativa. Por otro lado, una manera de que el usuario pueda reproducir cómodamente sus composiciones producidas en un contexto de Realidad Aumentada en un dispositivo con cámara como podría ser un *smartphone*.

Este escenario presenta ciertos retos de diseño e implementación. El software profesional de modelado profesional es complejo por una razón: tienen muchísimas posibilidades. Si se quiere tener una herramienta sencilla inevitablemente el usuario perderá opciones. Por tanto, se debe encontrar un compromiso, un *sweetspot* en el que se intenten cubrir casi todas las necesidades que podría tener una persona sin que el software se vuelva demasiado obtuso. También se debe de tener en cuenta un aspecto técnico importante: qué formato de archivos para modelos soportará el sistema. Existe una gran variedad hoy en día, cada uno con sus peculiaridades, y no siempre compatibles entre ellos, por lo que el software deberá adoptar uno o contar con herramientas de conversión para soportar un conjunto de ellos. También hay que tener en cuenta que el tipo de dispositivo en el que principalmente se tiene interés por reproducir escenas *RA* son teléfonos móviles o tablets debido a sus cámaras integradas y su portabilidad, pero estos se controlan con gestos táctiles, los cuales no son tan precisos como un ratón de ordenador y podría ser una dificultad a la hora de implementar controles precisos para el editor. Por ello deberá buscarse una forma de adaptar este tipo de programa a un control táctil o bien implementar una conectividad ordenador-*smartphone* para poder crear la escena en el primero y reproducirla en el segundo. En este TFG se propondrá y desarrollará una solución para estas necesidades.

2.2. Objetivos

Una vez definidos los retos y problemas a resolver en este contexto, se procede a enunciar los objetivos a los que se pretende llegar con este proyecto:

- **OBJ-1:** Desarrollar un software que permita cargar y crear modelos 3D a los que aplicarles transformaciones como translaciones, rotaciones y escalado para construir una escena.

- **OBJ-2:** El software descrito descrito debe ser simple, sencillo e intuitivo para que lo pueda usar cualquier persona sin que tenga experiencia ni conocimientos previos en la materia.
- **OBJ-3:** El software debe permitir la máxima flexibilidad posible a la hora de producir las escenas teniendo en cuenta su alcance limitado y controles sencillos.
- **OBJ-4:** Desarrollar una aplicación informática que reciba como entrada uno o varios modelos 3D que compongan una escena y reproduzca con ellos una experiencia de realidad aumentada.
- **OBJ-5:** La solución aportada debe ser ligera y lo más rápida posible, para que los usuarios de dispositivos de gama media o baja puedan usarlo sin problemas.
- **OBJ-6:** El sistema debe tener memoria, es decir, un usuario deberá de alguna forma poder conservar las escenas generadas con anterioridad para que el usuario las pueda volver a modificar en un futuro.

2.3. Estado del arte

Existe en el mercado cierta variedad de opciones desarrolladas que cumplen en mayor o menor medida los objetivos que se han planteado en este TFG. Esta sección se dedicará a analizarlas, compararlas y detectar cuales son las fortalezas y carencias de cada una, con la intención de sacar en claro qué se podría aportar con una nueva solución. Después de investigar y analizar las soluciones más relevantes, se encontraron **cinco plataformas** con un propósito y características similares al proyecto que se pretende desarrollar. De estas cinco se descartaron dos, *Aryel*³ y *Artificio*⁴ ya que aunque superficialmente comparten algunos elementos como son al fin de al cabo la creación de escenas *RA*, la primera está enfocada para que se le dé uso en agencias de marketing para diseñar publicidad interactiva, y la segunda es una herramienta para diseñar interiores de viviendas.

2.3.1. Onirix Studio

Onirix Studio [?] es una plataforma gratuita en la que un usuario al darse de alta puede crear multitud de escenas *RA* a través de un editor. Las escenas producidas quedan registradas en la cuenta de usuario, y pueden ser accedidas a través de un menú que permite volver a editarlas o eliminarlas. En cuanto al editor

³<https://aryel.io/>

⁴<https://artificio.com>

tiene un gran abanico de opciones. El usuario puede añadir modelos a la escena, posicionarlos, rotarlos y escalarlos con total libertad a través tanto de controles gestuales de ratón como de un panel numérico. Cuenta con la posibilidad de modificar la iluminación que proyecta la escena sobre los modelos. Es posible añadir texto que se visualice junto a los modelos. Para añadir *assets* como los modelos, el usuario cuenta con un catálogo personal, con algunos elementos que añade la aplicación por defecto, pero con la opción de que el usuario añada los suyos propios. Una vez los añade los puede usar en cualquier otro proyecto.

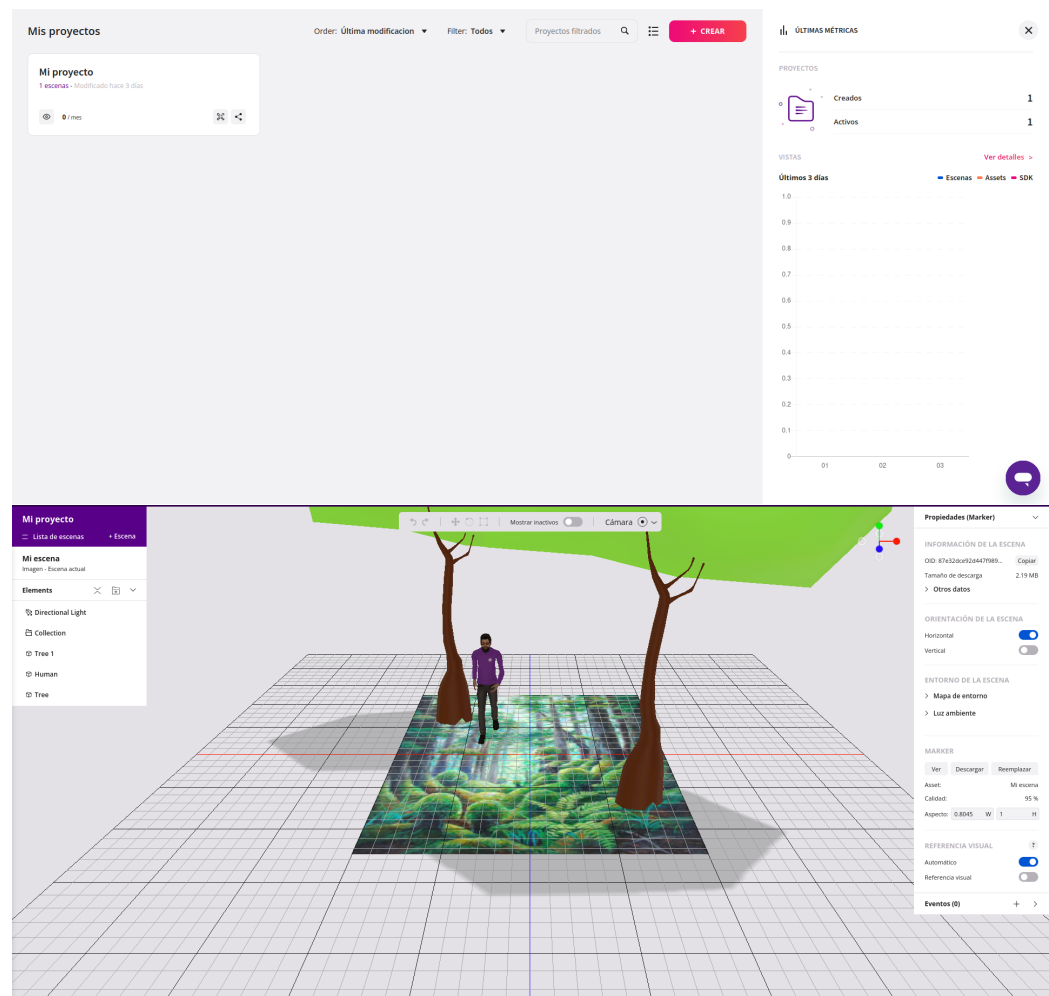


Figura 2.1: Web de creación de escenas RA Onirix Studio

Uno de los puntos más destacables que ofrece este entorno es un sistema al través del cual se pueden definir eventos dinámicos para la escena al cumplirse cierto activador. Por ejemplo, si tenemos un botón pulsable en la escena, se puede añadir una regla que ordena al pulsarse el botón la reproducción de la animación de uno de los modelos, o aparezcan modelos nuevos, desaparezcan

otros, etc.

Para la reproducción tenemos se pueden o bien reproducir en la propia web, o generar un *código QR*⁵ escaneable por un *smartphone*. Este *QR* abre en el navegador un enlace a través del cual se reproduce la escena. Por un lado, esta es una solución cómoda y ágil, pero tiene puntos negativos. Para empezar el rendimiento de la cámara y renderización de elementos 3D de un dispositivo móvil se va a ver reducida si es a través de una aplicación web en lugar de una aplicación nativa del teléfono. También hay que tener en cuenta que existe un enorme catálogo de navegadores gratuitos, y aunque la mayoría de los usuarios utilizan los más populares como *Google Chrome*, *Mozilla Firefox*, *Opera* o *Safari*, no se puede asegurar la compatibilidad total con cualquier navegador, o que el rendimiento sea óptimo en todos. Otro punto negativo a destacar de *Onirix Studio* [?] es que solo es posible crear escenas de posicionamiento sobre superficie y de imágenes aumentadas.

2.3.2. Blippbuilder

Blippbuilder [?] es una web muy similar a la anteriormente comentada, y también se puede emplear sin coste alguno. A través de una cuenta el usuario puede almacenar una colección de escenas RA editables a través de un menú similar. El editor contaba con una estructura parecida, permitiendo también aplicar transformaciones con gestos de ratón. Al igual que la anterior, tiene la opción tanto de cargar recursos desde el ordenador del usuario como de acceder a una librería incluida con la web, opción de añadir texto, manipular la iluminación de la escena. También cuenta con un pre visualizador dentro del editor y con opción para probar la escena desde el navegador de un dispositivo móvil escaneando un código *QR* (es decir, tampoco es nativa).

Como aspecto diferenciador, es posible añadir a la escena geometrías básicas tales como esferas, cubos o cilindros para añadirlos. Además, cuenta con un menú a través del cual se pueden **crear animaciones** para la escena. A través de *keyframes*. Un *keyframe* o *fotograma clave* es una unidad de información que almacena atributos tales como la posición, rotación o escala de un objeto para un instante determinado. Definiendo dos *keyframes* para el mismo objeto en 2 instantes distintos, un software puede interpolar sus posiciones intermedias generando así una animación. Esta animación puede ser reproducida junto a la escena.

En *Blippbuilder* solo es posible la creación de escenas de Realidad Aumentada por superficie.

⁵explicar código qr

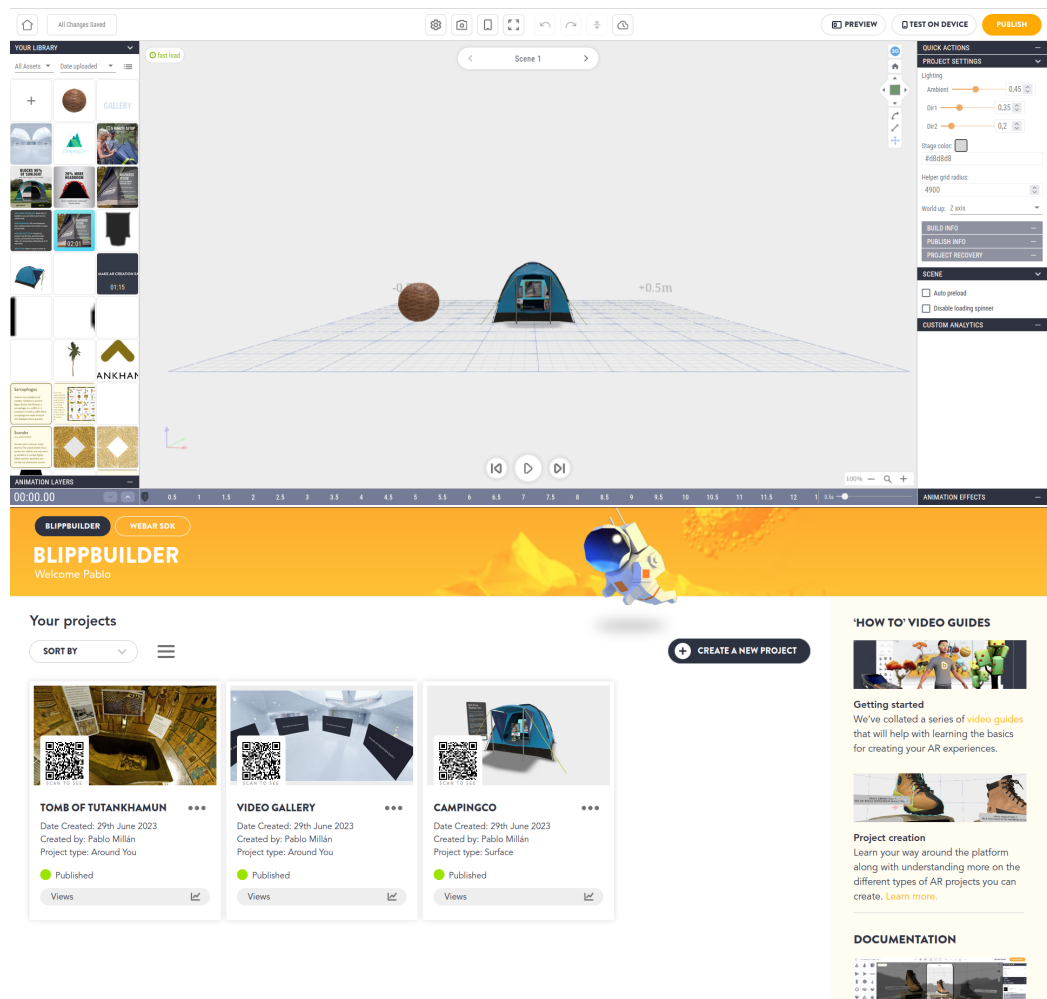


Figura 2.2: Web de creación de escenas RA Onirix Studio

2.3.3. Pictarize

Pictarize [1] de nuevo tiene un funcionamiento muy parecido a lo visto hasta ahora y sin requerir de pagos. Una galería de escenas producidas asociadas a una cuenta de usuario, y un editor para hacer las mismas. Tiene opción para cargar modelos desde el ordenador del cliente y de nuevo controles gestuales con el ratón para manipularlos. Aunque también puede añadir texto, no existe opción para alterar la iluminación. Sin embargo, es posible embeber videos de youtube en forma de textura plana, para reproducirlos al pulsarlos cuando se reproduce la escena. Es posible también asociar una pista de audio a la escena para que se reproduzca junto a las animaciones de los modelos 3D.

Para la reproducción de escenas se cuenta tanto con un pre visualizador

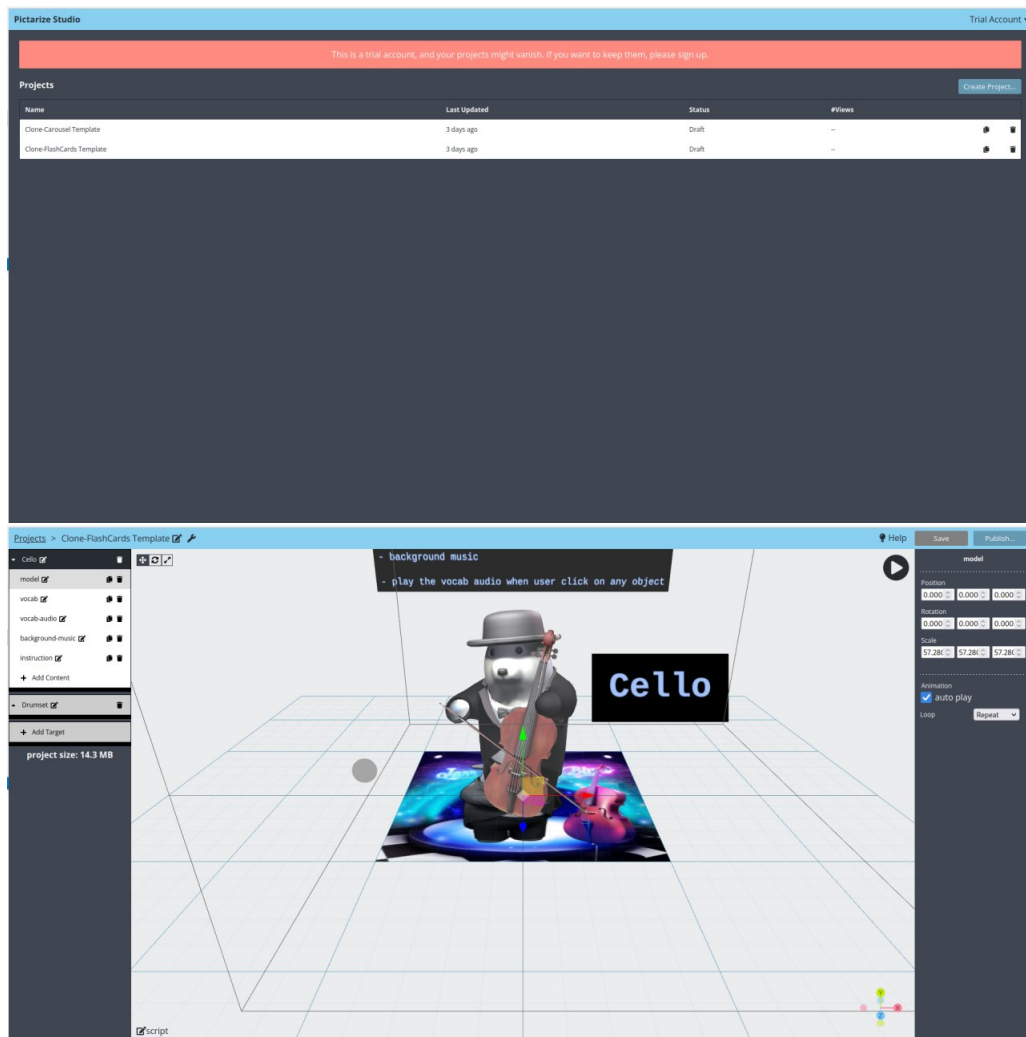


Figura 2.3: Web de creación de escenas RA Onirix Studio

dentro del editor como una opción para verlas desde un *smartphone* a través de un código *QR* y un navegador web, como se ha visto en los dos ejemplos anteriores. Sólo hay opción para escenas de imágenes aumentadas, con la opción de definir distintos activadores para distintos elementos.

2.4. Crítica

Como se ha podido dilucidar a través de estos ejemplos ya hay varias soluciones disponibles y bastante completas aun siendo gratuitas. Sus editores son potentes en cuanto a que permiten un alto grado de expresión por parte del

usuario, permitiendo un gran abanico de posibilidades de forma que el usuario podrá muy probablemente representar la escena que tenía en mente siempre que cuente con los modelos adecuados.

Las propuestas existentes analizadas comparten multitud de elementos en común. Todas ofrecen un menú para gestionar una lista de escenas creadas por un usuario, el cual se identifica con una cuenta para la propia web. Esta cuenta usa como identificador un nombre de usuario y correo electrónico. En todas es posible cargar objetos 3D desde el ordenador del cliente.

También permiten aplicar transformaciones de translación, rotación y escalado a los modelos de la escena con gestos del ratón a través de *gizmos*. Estos son unas estructuras visuales interactivas que ayudan a aplicar transformaciones a objetos 3D. Dependiendo de la operación que se quiera realizar, hay distintos tipos de gizmo. Cogiendo el ejemplo del *3D Move Gizmo* de la siguiente figura, si se hiciera click en una de las flechas coloreadas, se permitiría mover el objeto con un desplazamiento del ratón en el eje que señala la misma. Además de controles gestuales permiten introducir manualmente números para representar las transformaciones del objeto a través de un menú. Por ejemplo se puede especificar que un modelo está '*6.2 unidades desplazado en el eje X*' o con *una rotación de 90° en el eje Y*.

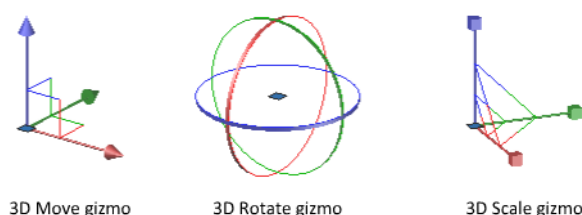


Figura 2.4: Ejemplo de Gizmo en la documentación de Autodesk

Hay otras funciones que si bien no son imprescindibles para el objetivo de crear una escena, dan posibilidades extra de personalización como la adición de texto, manipulación de luces, reproducción de audios, definición de eventos o creación de animaciones con *keyframes*. Hay que tener una cosa en cuenta con este tipo de funcionalidades, y es que si bien aportan valor y posibilidades a la aplicación, también aumentan la complejidad de la aplicación y la curva de aprendizaje que tienen los usuarios. No es descabellado pensar que alguien que no tiene nociones básicas sobre cómo funciona la animación tenga dificultades para manipular una timeline de *fotogramas clave* para crear una pequeña animación. Por tanto, se puede deducir que un porcentaje de los usuarios se sentirán confundidos por funciones como esta o la de definir eventos, o directamente no llegarán a intentar usarla.

También hay presentes algunas carencias en estas webs. Se puede ver que una de ellas permite crear tanto escenas por superficie como de imágenes aumen-

tadas, mientras que en las otras dos opciones solo es posible hacer o imágenes aumentadas o superficie. Y lo que es más, ninguna tiene soporte para generar escenas geoespaciales basadas en la localización.

Otra ausencia a destacar es la imposibilidad en ninguna propuesta de descargar guardar de forma local en el ordenador las escenas que se han creado en un formato de objeto 3D, para que el usuario pueda utilizar ese archivo en cualquier otro software que admita el tratamiento de modelos. Esto se traduce en una imposibilidad de que las escenas salgan del ecosistema de sus respectivas webs.

Por último señalar que en todas las webs analizadas la única opción para reproducir las escenas en un entorno real de Realidad Aumentada en un dispositivo móvil es a través de un navegador web al que se accede escaneando un código QR. Como se ha explicado anteriormente, se trata de una opción cómoda en tanto que es multiplataforma, no hace falta iniciar sesión de nuevo en el segundo dispositivo y en pocos segundos se puede escanear el código y tener cargando la escena, pero el rendimiento y fluidez de la cámara que se obtiene es considerablemente menor que si estuviera corriendo en una aplicación nativa, por lo que no estaría de más tener la opción de ejecutarlo de esta forma.

2.5. Propuesta

Con esta propuesta de proyecto se pretende desarrollar una alternativa completa que intentará cubrir las carencias que se encontraron analizando las webs disponibles en el mercado:

- **Intuitivo:** El editor contará con las herramientas básicas e imprescindibles para poder elaborar una escena 3D con las necesidades del usuario. Esto permitirá mantener una interfaz limpia, legible e intuitiva que permita construir escenas de una forma rápida y que no requiera conocimientos o experiencias previas con software de manipulación de entornos 3D. Así los usuarios no se sentirán perdidos o que tiene más herramientas de las que realmente necesitan.
- **Múltiples tipos de escenas:** Será posible la creación de distintos tipos de escenas de Realidad Aumentada. En concreto permitirá escenas de posicionamiento por superficies, de imágenes aumentadas y geoespaciales en unas coordenadas concretas.
- **Descarga de escenas:** Además de poder almacenar las creaciones en la nube, será posible exportar la escena con formato de objeto 3D para que el usuario pueda emplearla en cualquier otro software que permita archivos de este tipo.

- **Reproducción nativa:** Será posible reproducir escenas en una aplicación nativa para dispositivos móviles, proporcionando un mayor rendimiento y asegurando compatibilidad.

Para eso se desarrollará un servicio completo que contará de tres partes fundamentales. Para empezar, una **aplicación web** que permita la creación y gestión de escenas de realidad aumentada. Por otro lado, se tiene una **aplicación móvil** desde la que reproducir de forma nativa las escenas. Para terminar, un **servidor web** en el que se almacenarán las escenas creadas, que podrán ser recuperadas tanto desde la aplicación web como la aplicación móvil.

En el capítulo 4 se dará una hablará más en detalle del software en una descripción del mismo desde el ángulo de las metodologías ágiles, y en el capítulo 5 se explicará tanto la implementación como las decisiones de diseño y tecnologías empleadas.

Capítulo 3

Estado de la Realidad Aumentada

El software libre y sus licencias [4] ha permitido llevar a cabo una expansión del aprendizaje de la informática sin precedentes.

Capítulo 4

Planificación

4.1. Metodología utilizada

El software es un sistema complejo de crear que depende de muchas partes interconectadas. No es igual de dificultoso programar un pequeño *script* que ordene los ficheros de un directorio de un sistema operativo que construir una aplicación de cierta envergadura que responda a las necesidades de un usuario o un cliente concreto, mucho más si además se debe coordinar un equipo de personas en la concepción del mismo. Existe un gran factor de riesgo e incertidumbre, sobre todo al inicio de un proyecto. Las **metodologías de desarrollo ágil** son una forma de desarrollar software enfocada a equipos pequeños que busca paliar estos problemas a la hora de encarar un proyecto,. En otras metodologías anteriores se redactaban unos requisitos completos y fijos al inicio del proyecto y antes si quiera de comenzar la implementación. Esto era una fuente común de conflictos, ya que es muy común que los requisitos cambien a lo largo del desarrollo. Quizás el cliente cambie de idea, o el equipo encuentre mejores formas de realizar una tarea, o el contexto en el ámbito en el que se desarrolla el producto ha cambiado y hay que adaptar ciertas partes del proyecto, etc. La incertidumbre es un elemento muy presente, y las metodologías ágiles la abraza. Conviven con el cambio inevitable estableciendo unas bases menos sólidas pero más flexibles. Esta forma de trabajar se enunció oficialmente por primera vez en 2001, en el documento de **Manifiesto Ágil**¹. Esta forma de organizar proyectos se fundamenta en cuatro puntos, extraídos del libro *Métodos Ágiles y Scrum* [2]:

- **Valorar a individuos y sus interacciones**, frente a procesos y herramientas. Aunque todas las ayudas para desarrollar un trabajo son importantes,

¹El Manifiesto Ágil se puede consultar en esta web oficial. Hay una versión en español en la misma web. <https://agilemanifesto.org/>

nada sustituye a las personas, a las que hay que dar toda la importancia y poner en primer plano.

- **Valorar más el software (producto) que funciona**, que una documentación exhaustiva. Porque había llegado un punto en el que documentar el trabajo había alcanzado tanta importancia como el objeto del trabajo: el producto.
- **Valorar más la colaboración con el cliente** que la negociación de un contrato. La forma más productiva de sacar adelante un trabajo es establecer un marco de confianza y colaboración con quien nos lo encarga. Sin embargo se estaba poniendo el foco en cerrar un contrato atado que sirviera ante todo como una herramienta de protección, como si el cliente y equipo fueran dos partes enfrentadas, cuando en realidad comparten objetivos e intereses.
- **Valorar más la respuesta al cambio** que el seguimiento de un plan. Se trata de apreciar la incertidumbre como un componente básico del trabajo, por lo que la adaptación y la flexibilidad se convierten en virtudes y no en amenazas. El seguimiento ciego de un plan lleva, salvo contadas excepciones, al fracaso si no se puede corregir la dirección ante los inevitables cambios que van surgiendo.

Este tipo de metodologías intentan aplicar un cambio de paradigma: mientras que tradicionalmente se fijaban unos requisitos a partir de los cuales se estimaban los recursos y plazos que se requerirían para su terminación, la forma ágil sería fijar unos recursos y fechas disponibles en las que se disponen para el trabajo, y a partir de ahí estimar los requisitos que va a tener el producto al final. Se eligió optar por esta forma de encarar proyectos ya que en primer lugar el trabajo tenía una fecha de finalización fija, y además no se tenía experiencia en algunos de los campos requeridos para el desarrollo, como podría ser por ejemplo la implementación de interfaces web. Se iba a requerir de un periodo de formación paralela al desarrollo, lo cual con total probabilidad cambiará la visión del desarrollador sobre la estimación de los costes y tiempo del proyecto. Sería contraproducente espablecer desde un inicio unos requisitos fijos si se prevé que van a cambiar, y las metodologías ágiles se ajustaban perfectamente a este escenario.

Una vez definidos los métodos de desarrollo ágil y entendido por qué son convenientes para este proyecto en particular, se debe elegir uno de los que hay disponibles como *Lean Software Development*, *Kanban*, *Pragmatic Programming* o *eXtreme Programming*. En este caso se ha optado por **SCRUM** [10], una de las metodologías más difundidas.

SCRUM proporciona un marco de trabajo para equipos de desarrollo pequeños y autogestionados. Como en este caso el trabajo se realiza de forma

individual los roles que propone este método se han sintetizado en una sola persona. Originalmente se tenía al equipo de desarrollo, el *Product Owner* (dueño del producto o cliente) y el *SCRUM Master*, el encargado de que el equipo sea efectivo.

Se trabaja definiendo una serie de *sprints*, iteraciones de dos semanas de duración las cuales deben presentar un producto entregable al final de cada una, como por ejemplo un software ejecutable. Al comienzo de cada sprint se hace el *sprint planning*, una reunión en la que se planifica el trabajo a desarrollar durante la iteración. Al final de cada sprint se realiza el *sprint review* en el que se valora cómo se ha trabajado y qué aspectos mejorar para el siguiente sprint. Las *daily meetings* que propone SCRUM en las que cada miembro del equipo comenta en qué estado del trabajo se encuentra se ha omitido para este proyecto por ser solo un integrante en el equipo.

4.2. Historias de usuario

Para describir los requisitos del sistema a desarrollar se emplearán *historias de usuario*. Estas son descripciones del sistema que se usan para plasmar los requisitos del mismo desde el punto de vista del usuario. Se redactan al comienzo del desarrollo, durante el *Sprint 0* del que se hablará más adelante, y a partir de estas se redactan las *tareas* del desarrollo.

Antes de redactar las historias es necesario definir los distintos perfiles que van a hacer uso del producto. En este caso se tiene un único perfil:

- **Usuario:** Este perfil representa a un consumidor casual, que puede o no tener nociones tanto de Realidad Aumentada como de modelado 3D. Tiene una experiencia como poco básica en el manejo de ordenadores y aplicaciones de los mismos, usando tanto el teclado como el ratón para interactuar con distintos elementos en pantalla. Es usuario de servicios como correo electrónico y está familiarizado con el proceso de iniciar sesión con contraseña. Tiene los recursos para poder encontrar en internet modelos 3D gratuitos y descargarlos en su equipo.

Una vez definido el perfil que hará uso del software se redactarán los *journeys*. Estas son una descripción en forma de narración en la que se detalla el uso que un usuario concreto (que no un perfil de usuario) hace uso del producto desde el inicio hasta al final. Esto servirá para comprender mejor las necesidades de estos y reflejarlas en las historias de usuarios.

- **J-01:** Florentino Pérez es un hombre de 42 años. Escucha sobre la tecnología de Realidad Aumentada en el telediario y le parece de lo más vistosa e innovadora, por lo que suscita su curiosidad y decide probarla. Tras una

breve búsqueda en Google encuentra EditAR entre otras opciones, y decide probarla porque aparenta ser sencilla. Tras acceder a la web sigue las instrucciones para registrarse como usuario. Una vez hecho, selecciona la opción de crear escena, y empieza a componer una del tipo de posicionamiento en superficies. Carga modelos 3D que ha encontrado en una web de recursos gratuitos y hace click en ellos para manipularlos y colocarlos donde él prefiera. Cuando ha terminado selecciona la opción de guardar la escena. Florentino procede a instalar en su dispositivo móvil la aplicación del visor. En ella inicia sesión con las credenciales que introdujo en pasos anteriores. Aparece un menú con la única escena que ha creado y selecciona la opción de reproducir. Se inicia la cámara de su dispositivo y procede a colocar el modelo de la escena encima de su escritorio. Florentino observa cómo detecta la superficie y coloca el objeto. Incluso aunque mueva la cámara de ángulo sigue dando la sensación de que el objeto está en la mesa. Satisfecho, cierra la aplicación.

- **J-02:** Belkan Díaz es un hombre de unos 20 años. Acaba de adquirir un nuevo y flamante smartphone Xiaomi y está oteando la tienda de aplicaciones para descargar algunas para estrenarlo. De casualidad se topa con la aplicación de EditAR y acaba descubriendo también su web, así que decide probarla. Se registra en el sistema con sus credenciales y pasa a crear una escena de imágenes aumentadas con modelos 3D gratuitos de internet. Cuando ha terminado y la guarda, se da cuenta de que por error se ha registrado con el correo electrónico de su trabajo, lo cual es inapropiado, así que accede al menú de cambio de credenciales, e introduciendo su contraseña logra cambiar su dirección de email. Una vez solucionado eso, accede usando su nuevo correo desde la aplicación móvil y consigue reproducir la escena. Aunque le sorprende el funcionamiento de la realidad aumentada, no le gusta la escena que ha creado para nada, así que vuelve al editor para eliminarla y crear una nueva, la cual sí le convence.
- **J-03:** Antonia Miyamoto es una artista de modelos 3D. Tras haber oído hablar de la aplicación decide probarla con alguno de los modelos que ella misma ha diseñado. Accede a la web y se registra con sus credenciales. Piensa en que le gustaría ver un dragón sobrevolando su casa, así que inicia una nueva escena y la configura para que sea de tipo geoespacial. Introduce las coordenadas y altura de su vivienda, que ha podido obtener en algún servicio como *Google Earth*. Carga el modelo del dragón que modeló días anteriores en *Blender*. Lo coloca en la escena y selecciona que quiere que se reproduzca una animación del modelo en la que el dragón bate sus alas, para que de la sensación de que verdaderamente está volando. Una vez se da por satisfecha guarda la escena y cierra el editor. Instala la aplicación móvil e inicia sesión. Selecciona su escena y ve al dragón volando. Al verlo, piensa que podría quedar mucho mejor si mete una gran bola de fuego y

una pista de audio de un gruñido de dragón. Antonia vuelve a abrir el editor y selecciona la escena que ya había creado en una opción que le permite editarla. El editor carga la escena tal como la dejó, lo que le permite añadir nuevos modelos para añadirle fuego. Además carga un archivo mp3 con el rugido. Cuando ve la nueva escena en su dispositivo móvil piensa que la escena ha quedado verdaderamente vistosa, y le gustaría exportarla para usarla en alguno de sus proyectos de *Blender*, así que vuelve a abrir el editor web con la escena creada y hace click en un botón que le exporta su escena y la descarga a su ordenador en forma de modelo 3D.

Una vez entendidas mejor las necesidades que tendrá el usuario del sistema se pueden redactar las historias de usuario. Para cada historia se definen una id que servirá para identificarla con formato **RA-X**, una descripción, unos requisitos asociados con formato **RF/RNF-X.n** y una prioridad. La prioridad podrá ser *high* (alta) si describen elementos imprescindibles para el correcto funcionamiento del software y el cumplimiento de su propósito, *mid* (media) si son de gran importancia pero la ausencia de alguna no es un impedimento crítico para el producto final, y *low* para las historias que describen elementos preferibles, que aportan valor añadido pero que no son necesarios.

4.2.1. RA-01: Entorno 3D

Como usuario quiero tener un entorno 3D interactivo para configurar mis escenas de Realidad Aumentada.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.2. RA-02.1: Escenas de superficie

Como usuario la posibilidad de crear escenas de Realidad Aumentada de posicionamiento por superficies.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.3. RA-02.2: Escenas de imágenes aumentadas

Como usuario la posibilidad de crear escenas de Realidad Aumentada de imágenes aumentadas.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.4. RA-02.3: Escenas geoespaciales

Como usuario la posibilidad de crear escenas de Realidad Aumentada geoespaciales introduciendo una latitud, longitud y altura concreta.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.5. RA-03: Varias imágenes activadoras

Como usuario quiero, al crear una escena por marcador, poder definir varias imágenes activadoras, y distintos modelos asociados a cada una de ellas.

- Prioridad: **LOW**
- Requisitos: Ninguno

4.2.6. RA-04: Animaciones

Como usuario quiero que los modelos que cargo puedan tener animaciones, y decidir si se reproducen al ver la escena una vez, en bucle o no lo hacen.

- Prioridad: **MID**
- Requisitos: Ninguno

4.2.7. RA-05: Pistas de audio

Como usuario quiero poder cargar audios que se reproduzcan cuando se dispara la escena, y decidir si se reproducen una vez o en bucle.

- Prioridad: **MID**
- Requisitos:
 - RF-5.1: Cada escena tendrá asociado un único audio como mucho.

4.2.8. RA-06: Carga de modelos

Como usuario quiero cargar un modelo 3D desde mi ordenador para añadirlo a la escena, así como eliminarlo más adelante.

- Prioridad: **HIGH**
- Requisitos:
 - **RNF-6.1:** Los modelos cargados deben ser de formato GLB con las restricciones para su uso en Filament.

4.2.9. RA-07: Transformaciones

Como usuario quiero poder seleccionar un objeto 3D de la escena haciendo click sobre él y poder moverlo, rotarlo o hacerlo más grande o más pequeño.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.10. RA-08: Selección múltiple

Como usuario quiero poder seleccionar más de un objeto a la vez para poder aplicarles los mismos cambios como moverlos, rotarlos o cambiarlos de tamaño.

- Prioridad: **LOW**
- Requisitos: Ninguno

4.2.11. RA-09: Figuras básicas

Como usuario quiero poder crear figuras geométricas básicas como esferas, cubos, pirámides o cilindros entre otras.

- Prioridad: **LOW**
- Requisitos: Ninguno

4.2.12. RA-09.1: Texturas

Como usuario quiero un menú en el que poder cambiar el color de las figuras básicas que creo.

- Prioridad: **LOW**
- Requisitos: Ninguno

4.2.13. RA-10: Inicio de sesión

Como usuario quiero poder iniciar sesión con mis datos en la aplicación web para que, si accedo desde otro ordenador, pueda recuperar datos míos como las escenas creadas.

- Prioridad: **HIGH**
- Requisitos:
 - **RNF-10.1:** Se almacenará y empleará para el inicio de sesión únicamente el email y contraseña del usuario

4.2.14. RA-10.1: Cambio de credenciales

Como usuario quiero tener la opción de modificar mi correo electrónico o contraseña en un futuro.

- Prioridad: **MID**
- Requisitos: Ninguno

4.2.15. RA-10.2: Guardado de escenas

Como usuario quiero poder guardar bajo mi cuenta de usuario las escenas que creo para poder volver a abrirlas en un futuro y editarlas.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.16. RA-11.1: Reproducción de imágenes aumentadas

Como usuario quiero poder reproducir en una app móvil en la que reproducir mis escenas de imágenes aumentadas, detectando con la cámara del teléfono la imagen que establecí en el editor como disparador de la escena.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.17. RA-11.2: Reproducción de escenas por superficies

Como usuario quiero poder reproducir en una app móvil en la que reproducir mis escenas por superficies, las cuales podré colocar en sitios como mesas o suelos.

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.18. RA-11.3: Reproducción de escenas geoespaciales

Como usuario quiero poder reproducir en una app móvil en la que reproducir mis escenas geoespaciales, en las cuales podré ver los modelos solo si estoy cerca de ciertas coordenadas que introduje en el editor a la hora de crear la escena.

- Prioridad: **MID**
- Requisitos: Ninguno

4.2.19. RA-12: Inicio de sesión en la app móvil

Como usuario quiero iniciar sesión en la app móvil para acceder a las escenas que previamente he creado

- Prioridad: **HIGH**
- Requisitos: Ninguno

4.2.20. RA-12.1: Listado de escenas

Como usuario quiero tener en la app móvil un listado de todas las escenas que he creado para poder seleccionarla y reproducirla.

- Prioridad: **MID**
- Requisitos: Ninguno

4.2.21. RA-12.1: Filtrado de escenas

Como usuario quiero en el listado de escenas poder filtrarlas por tipo o por nombre, para poder encontrar la que me interesa rápidamente.

- Prioridad: **LOW**
- Requisitos: Ninguno

4.2.22. RA-13: Exportación de escenas

Como usuario quiero poder exportar mi escena y modelos a un formato que permita ser utilizado por aplicaciones móviles de realidad aumentada.

- Prioridad: **HIGH**
- Requisitos:
 - **RNF-13.1:** Las escenas con varios modelos cargados deben exportarse como un solo modelo.

4.3. Sprint 0

El sprint 0 tiene la el objetivo de definir todo el trabajo que se va a realizar y cómo. Se estudia el problema para saber si es viable construir una solución. Esta iteración especial tuvo una duración de una semana. Aquí se realizaron las labores de investigación que se ha perseguido en los capítulo 2 y 3 de este trabajo. También se crea el *product backlog*, la lista de historias de usuario que recoge la visión de los requisitos principales de la aplicación desde la perspectiva de sus usuarios que ya se redactó en el apartado anterior.

Otra actividad esencial de esta etapa es la definición y temporización de los sprints que se desarrollarán en todo el desarrollo. Se decidió establecer un total de **cinco sprints de dos semanas cada uno** (sin incluir este). Las historias de usuario se dividirían entre los distintos sprints. Para cuando acaben cada uno de ellos, debe resultar una versión entregable del producto que implemente las funcionalidades y requisitos especificados por sus historias de usuario. Para ello, de cada historia de usuario se extraerá al principio de cada sprint una serie de **tareas** que completar para avanzar el desarrollo. Se dividieron las historias de tal forma que las que se encasillen en un mismo sprint, guarden cierta relación entre ellas tematizando así las labores que se desarrollan en cada iteración. También se tuvieron en cuenta la prioridad de las historias, dejando la mayoría de las menos relevantes para sprints finales, por si hubiera complicaciones en alguno de los anteriores, priorizar las funcionalidades más esenciales.

Se definió también la temporización de estos sprints, comenzando el desarrollo el 17 de abril de 2023 y finalizando el 25 de junio de 2023:

- **Sprint 1:** 17/4/23 a 30/4/23
- **Sprint 2:** 1/5/23 a 14/5/23
- **Sprint 3:** 15/5/23 a 28/5/23
- **Sprint 4:** 29/5/23 a 11/6/23

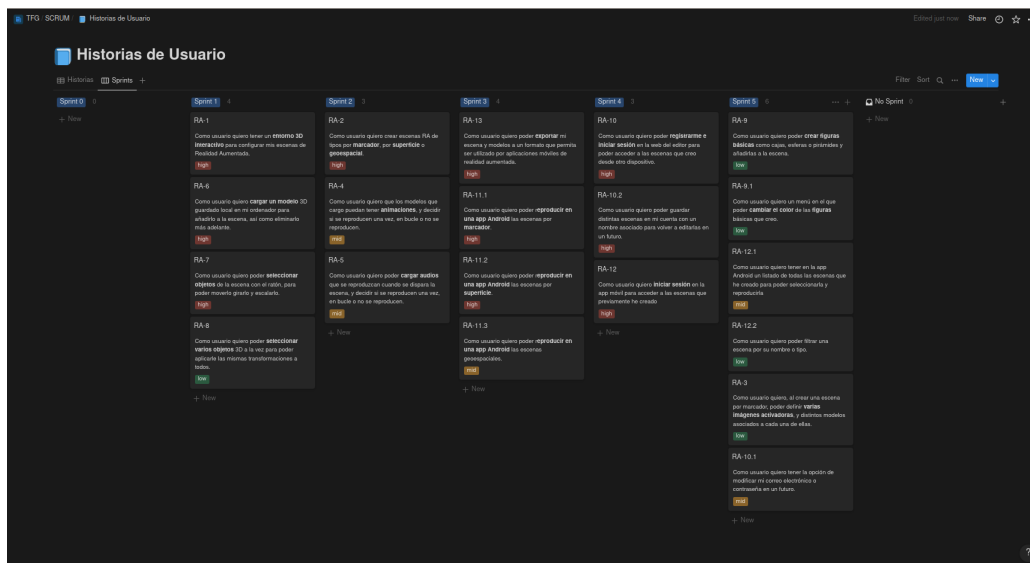


Figura 4.1: Distribución de las historias de usuario por sprints

■ **Sprint 5:** 12/6/23 a 25/6/23

Para cada sprint se tendrá un tablero de tareas en el que de un vistazo el desarrollador podrá obtener una idea de en qué estado del desarrollo se encuentra. Para ello se organizará la tabla en columnas, bajo las que se colocarán las distintas tareas según su estado, que pueden ser:

- **Backlog:** Listado de tareas por implementar en el sprint
- **Implementación:** Las tareas que están en proceso de implementación en este momento.
- **Terminada:** Tareas finalizadas por completo.
- **Diseño:** Tareas que están en fase de diseño porque requieren de la elaboración de diagramas o similares.
- **Investigación:** Lista de tareas que requieren alguna labor de investigación adicional que no se había previsto.
- **Bloqueado:** Tareas que no pueden ser implementadas hasta que finalicen otras o que se han descartado por alguna razón.

Otra tarea esencial del sprint 0 es definir las tecnologías y herramientas que se usarán para el desarrollo para tener una visión más amplia del producto final y reducir la incertidumbre del desarrollo, por lo que es lo que se va a proceder a describir.

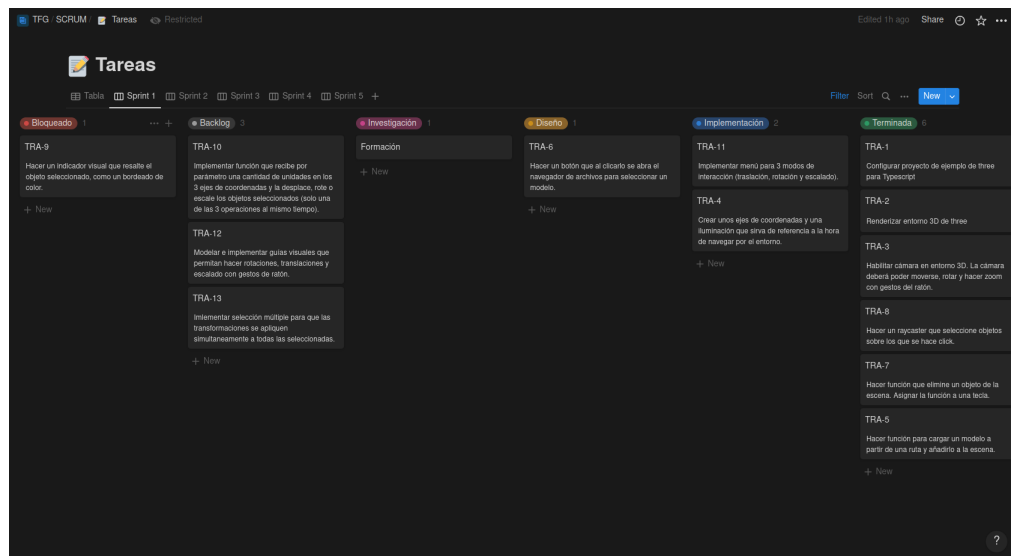


Figura 4.2: Ejemplo de tabla para el sprint 1

Como software de gestión de proyecto se optó por usar Notion [?]. Aquí se gestiona todo lo relacionado con la planificación del trabajo: historias, tareas, sprints, requisitos, tableros, etc. Se contemplaron otras opciones, principalmente *Jira* [?], pero era una herramienta demasiado compleja ya que está contemplada para el trabajo en equipo y en este caso el desarrollador era una única persona, por lo que era un software que venía demasiado grande para el propósito. Notion por otro lado, aunque no esté directamente diseñado para estos propósitos, permite la creación de tablas, bases de datos y tableros, hiperenlaces a distintos documentos dentro de un espacio de trabajo y almacenamiento en la nube. Como software de gestión de proyectos se empleó *Git* y *GitHub*, ya que es la plataforma más utilizada y con la que se contaba experiencia previamente, tenía todas las funcionalidades que se requerían para el proyecto. Se creó un repositorio² público desde el primer momento con el proyecto íntegro publicado bajo licencia de software libre.

En cuanto a desarrollar aplicaciones web se refiere, *JavaScript* es la única opción existente en cuanto a lenguajes ya que es el único que pueden ejecutar todos los navegadores, así que no hubo muchas dudas en ese aspecto. Aunque no se optó exactamente por *JavaScript*, si no por *TypeScript*, una extensión sintáctica para el lenguaje que añade la opción de introducir tipado estático a los objetos. Se tomó esta decisión porque se consideró que a cambio de invertir un poco más de tiempo en escribir el código, se podría ahorrar tiempo de depuración. Esto es posible ya que gracias a declarar los tipos de las variables, el editor puede detectar errores antes incluso de la ejecución, como por ejemplo intentar añadir

²<https://github.com/pabloMillanCb/tfg>

un objeto *String* a un array de objetos *Object 3D*, evitándose así sorpresas desagradables en la ejecución.

Como se pretende desarrollar un editor que muestre y trabaje con modelos 3D, se necesitará un motor de gráficos por ordenador operable desde *JavaScript* y con soporte para web. Las dos opciones que se tuvieron en cuenta fueron *Three.js* [?] y *Babylon.js* [?], dos librerías *JavaScript* para la renderización de gráficos 3D. Se optó por coger la primera, ya que tras analizar ambas no se encontró ninguna ventaja significativa de una respecto a otra, y para *Three.js* se contaba ya con experiencia, lo que aceleraría el desarrollo.

Para construir la aplicación web se barajaron dos opciones: *Angular* [?], un framework completo para *JS* para el desarrollo de web y móvil o *React.js* [?], una librería *JS* para el desarrollo frontend. Se escogió esta última ya que, aunque *Angular* es más completo y no necesita librerías externas para realizar ciertas funcionalidades, *React* es mucho más simple e intuitivo. Su curva de aprendizaje es mucho menos pronunciada, casi inmediata si se tiene conocimiento en *JavaScript* como era el caso. Como no se tenía experiencia en ninguna de las dos herramientas, se optó por esta para reducir el periodo de formación requerido.

Para el visor de las escenas en móvil se decidió hacer una aplicación Android. Se es consciente de el hecho que supone una desventaja respecto a la versión web de otras opciones del mercado ya que en dispositivos iOS no funcionaría, pero se decidió apostar por una aplicación nativa ya que no la ofrece ninguna otra propuesta. Se escogió Android porque es la plataforma mayoritaria en el mercado móvil. Para el desarrollo de la aplicación se escogió usar *Android Studio* frente a *React Native*. Ambos permiten escribir aplicaciones nativas, pero la primera tiene una mayor integración con *Sceneview*, la librería escogida para las funcionalidades de Realidad Aumentada. Se programará en Kotlin en lugar de Java, ya que es la recomendación oficial de Google y la forma más moderna de hacerlo.

Para las funcionalidades de Realidad Aumentada se optó por ARCore [6], el SDK de Google para esta tecnología. El único problema que se presentaba con esta decisión es que ARCore renderiza objetos 3D a través de OpenGL, una API de bajo nivel que ralentizaría bastante el desarrollo. Para paliarlo se adoptó Sceneview [8], una librería para Kotlin que envuelve a ARCore y hace el renderizado mucho más simple.

Para las necesidades del proyecto era evidente que haría falta algún tipo de base de datos para almacenar información como los usuarios, las escenas de estos y los recursos como modelos y audio que componen las escenas. Era necesario desarrollar una API REST, una interfaz que emplean dos sistemas informáticos para intercambiar información de manera segura a través de la web. Para ello se pensó en un primer momento hacerla con PHP y MySQL, pero finalmente se decidió usar *Firebase* [7]. Esta herramienta viene con muchas utilidades ya implementadas para que el desarrollador pueda hacer uso de estas con llamadas simples a su API, además de no invertir tiempo extra de desarrollo.

Trae incorporado un servicio para el registro e inicio de sesión de usuarios **con contraseña encriptada** de forma automática, almacenamiento para archivos binarios y una base de datos no relacional entre otros. Para lo sencillo que se previó que iba a ser la infraestructura del backend (solo se necesitarían tablas para usuarios y escenas, además de una forma de guardar binarios en la base de datos) era una opción perfecta que se acomodaba a las necesidades y reducía mucho la complejidad del desarrollo.

Si bien Firebase ofrece todos esos servicios, seguía haciendo falta una API que lo comunicara con los clientes web y Android. Se decidió usar *Node.js* [5] junto a *Express* [3], ya que es una dupla ampliamente usada para el desarrollo de aplicaciones con Firebase con muchos recursos disponibles en internet. Además, no habría que familiarizarse con lenguajes nuevos, pues se programaría en *JavaScript*

4.4. Sprint 1

Para este sprint se centraron esfuerzos en desarrollar las funcionalidades básicas del editor. El objetivo era que al final del sprint se tuviera un entorno 3D con una cámara la cual pudieras mover y rotar con el ratón, que hubieran varios objetos de prueba que fueran seleccionables con un click del ratón, además de la opción de cargar modelos. También se podrían aplicar transformaciones a través de un menú. Para realizar todo esto se añadieron al sprint la RA-01, RA-06, RA-07 y RA-08. De cada historia se extrae un backlog de tareas que se detalla a continuación.

Tarea: TRA-01	Horas estimadas: 3	Horas totales: 1	HU asociada: RA-01
Configurar proyecto de ejemplo en Three.js con TypeScript			

Tarea: TRA-02	Horas estimadas: 1	Horas totales: 1	HU asociada: RA-01
Renderizar entorno 3D de Three.js			

Tarea: TRA-03	Horas estimadas: 2	Horas totales: 2	HU asociada: RA-01
Habilitar cámara en entorno 3D. La cámara deberá poder moverse, rotar y hacer zoom con gestos del ratón.			

Tarea: TRA-04	Horas estimadas: 3	Horas totales: 0.5	HU asociada: RA-01
Crear unos ejes de coordenadas y una iluminación que sirva de referencia a la hora de navegar por el entorno.			

Tarea: TRA-05	Horas estimadas: 1	Horas totales: 0.1	HU asociada: RA-06
Hacer una función para cargar un modelo a partir de una ruta y añadirlo a la escena.			

Tarea: TRA-06	Horas estimadas: 1	Horas totales: 1	HU asociada: RA-06
Hacer un botón que al clicarlo se abra el navegador de archivos del navegador para seleccionar un modelo y cargarlo.			

Tarea: TRA-07	Horas estimadas: 1	Horas totales: 0.5	HU asociada: RA-06
Hacer función que elimine un objeto de la escena.			

Tarea: TRA-08	Horas estimadas: 2	Horas totales: 2	HU asociada: RA-07
Hacer un raycaster que seleccione objetos sobre los que se hace click.			

Tarea: TRA-09	Horas estimadas: 3	Horas totales: 0.5	HU asociada: RA-07
Hacer un indicador visual que resalte el objeto seleccionado, como un bordeado de color.			

Tarea: TRA-10	Horas estimadas: 2	Horas totales: 2	HU asociada: RA-07
Implementar función que recibe por parámetro una cantidad de unidades en los 3 ejes de coordenadas y la desplace, rote o escale los objetos seleccionados (solo una de las 3 operaciones al mismo tiempo).			

Tarea: TRA-12	Horas estimadas: 8	Horas totales: 8	HU asociada: RA-07
Modelar e implementar guías visuales que permitan hacer rotaciones, translaciones y escalado con gestos de ratón.			

Tarea: TRA-13	Horas estimadas: 1	Horas totales: 2	HU asociada: RA-08
Implementar selección múltiple para que las transformaciones se apliquen simultáneamente a todas las seleccionadas.			

El transcurso del sprint fue sin demasiadas complicaciones. Se terminaron todas las tareas a excepción de dos. La TRA-09 decidió cancelarse debido a que se pensó que en realidad el *gizmo* que se implementaría para transformar los objetos en sprints posteriores ya era suficiente indicador visual para saber que un objeto estaba seleccionado. La TRA-11 describía un menú para seleccionar los tres posibles modos de interacción con los objetos (translación, rotación y escalado). Se canceló debido a que a lo largo del sprint se pensó que sería mejor esperar a tener diseños para toda la interfaz e implementarlo todo de golpe, que hacer ahora un menú que con toda probabilidad sería descartado, así que simplemente cada funcionalidad se asignó a una tecla. La gráfica de burndown del sprint es la siguiente:

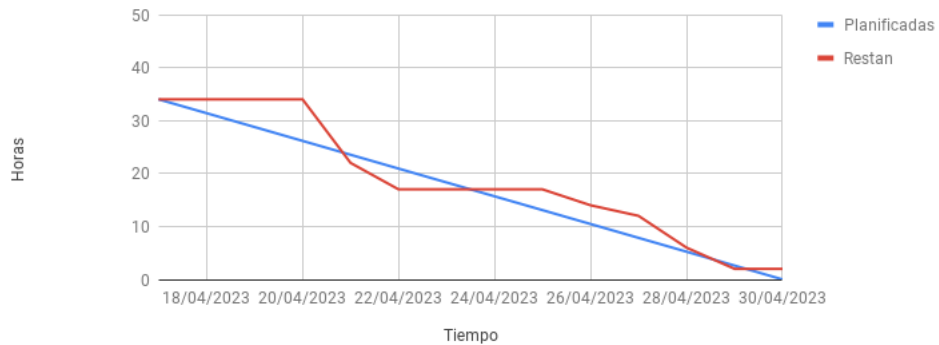


Figura 4.3: Gráfica de la progresión del Burndown en el Sprint 1

4.5. Sprint 2

En este segundo sprint se centró el trabajo principalmente en implementar toda la interfaz de usuario que correspondería a la página del editor, incluso si muchas de las funciones accesibles desde este no estaban implementadas. Esto requirió una formación continua a lo largo del sprint en React, tecnología para la que no se contaba experiencia hasta la fecha. Además se implementaría la reproducción de animaciones para los modelos cargados y de audios cargados a la escena. Las historias de usuario incluidas en este sprint son RA-02, RA-04 y RA-05.

Tarea: TRA-14	Horas estimadas: 3	Horas totales: 1	HU asociada: RA-02
Hacer diseños de la interfaz de usuario del editor.			

Tarea: TRA-15	Horas estimadas: 15	Horas totales: 14	HU asociada: RA-02
Implementar la interfaz de usuario en React			

Tarea: TRA-16	Horas estimadas: 2	Horas totales: 2	HU asociada: RA-04
Implementar función que reproduzca una animación en todos los objeto 3D de la escena si es que tienen.			

Tarea: TRA-17	Horas estimadas: 1	Horas totales: 0.5	HU asociada: RA-05
Implementar una función que cargue un audio desde un archivo seleccionado desde el navegador de archivos del sistema.			

Tarea: TRA-18	Horas estimadas: 1	Horas totales: 0.5	HU asociada: RA-05
Implementar una función que reproduzca el audio cargado de la escena.			

Tarea: TRA-19	Horas estimadas: 2	Horas totales: 4	HU asociada: RA-02
Asignar a cada elemento de la interfaz de usuario su funcionalidad correspondiente del editor si es que ya está implementada.			

Tarea: TRA-20	Horas estimadas: 2	Horas totales: 3	HU asociada: RA-02
Migrar el código del editor en Three.js a React.			

Tarea: TRA-21	Horas estimadas: 1	Horas totales: 1	HU asociada: RA-02
Hacer una página de placeholder que sea la inicial al abrir la aplicación web. Tendrá un único botón que llevará a otra página donde se encuentra el editor.			

Además, durante el transcurso del sprint se añadió una nueva tarea que no se había contemplado al inicio:

Tarea: TRA-22	Horas estimadas: 1	Horas totales: 0.5	HU asociada: RA-02
Mudar la lógica de control de la escena de EditorComponent.tsx a una clase Typescript.			

Para comentar el transcurso de este sprint debemos observar antes la gráfica del burndown.



Figura 4.4: Gráfica de la progresión del Burndown en el Sprint 2

Como se puede observar, la línea roja del progreso no llega a tocar el eje X de la gráfica, representando que efectivamente quedaron algunas tareas sin completar. Esto se achaca a un problema puntual de salud y de trabajo externo, no a una mala planificación o previsión. Se llega a este razonamiento porque, de las 28 horas planificadas, se realizaron **24,5 netas y 21,5 reales**. No se habían subestimado las tareas en sí, si no el tiempo que se tendría disponible para trabajar en el proyecto durante la franja de la iteración. Independientemente

de la razón, el proyecto estaba sufriendo un pequeño retraso, y había que paliarlo de alguna manera. Como opciones se presentaban alargar algunos días el sprint, o pasar las tareas sin completar a la siguiente iteración. Se optó por esto último. Las tareas TRA-21 y TRA-16 se terminarían en el Sprint 3.

4.6. Sprint 3

Este sprint orbitó alrededor del desarrollo de las funcionalidades básicas para la app Android. Se pretendía tener funcionales los tres tipos de escenas en esta. Además se implementaría la opción de exportar y descargar escenas desde el editor web. Por último se completarían las tareas atrasadas del sprint anterior. Las historias que describen este sprint son RA-13, RA-11.1, RA-11.2, RA-11.3.

Tarea: TRA-22b	Horas estimadas: 0.5	Horas totales: 0.5	HUs RA-11.1 RA-11.2 RA-11.3
Realizar configuración inicial del proyecto de Android Studio.			

Tarea: TRA-23	Horas estimadas: 2	Horas totales: 0.5	HUs RA-11.1 RA-11.2 RA-11.3
Implementar reproducción de sonidos en la app de Android Studio.			

Tarea: TRA-24	Horas estimadas: 2	Horas totales: 2	HUs RA-11.1 RA-11.2 RA-11.3
Implementar escena AR de ejemplo de la documentación de Sceneview.			

Tarea: TRA-25	Horas estimadas: 3	Horas totales: 2	HU asociada RA-11.1
Implementar modo de escenas por marcador en la app Android siguiendo el ejemplo de la documentación.			

Tarea: TRA-26	Horas estimadas: 3	Horas totales: 3	HU asociada RA-11.3
Implementar modo de escenas geoespaciales en la app Android siguiendo el ejemplo de la documentación.			

Tarea: TRA-27	Horas estimadas: 3	Horas totales: 2	HU asociada RA-11.2
Implementar modo de escenas de posicionamiento sobre superficies en la app Android siguiendo el ejemplo de la documentación.			

Tarea: TRA-28	Horas estimadas: 2	Horas totales: 0.5	HUs RA-11.1 RA-11.2 RA-11.3
Implementar la reproducción de animaciones para los modelos en caso de que estos tengan y así se especifique en el JSON de entrada.			

Tarea: TRA-29	Horas estimadas: 8	Horas totales: 6	HU asociada RA-13
Implementar una función que exporte todos los modelos de la escena a un solo modelo glb y lo descargue.			

Tarea: TRA-30	Horas estimadas: 0.5	Horas totales: 1	HU asociada RA-13
Diseñar un archivo JSON para la salida del editor y la entrada de la app. Este deberá indicar los parámetros necesarios para la configuración de una escena, como el tipo de la misma, si tiene audio o animaciones, sus modelos, etc.			

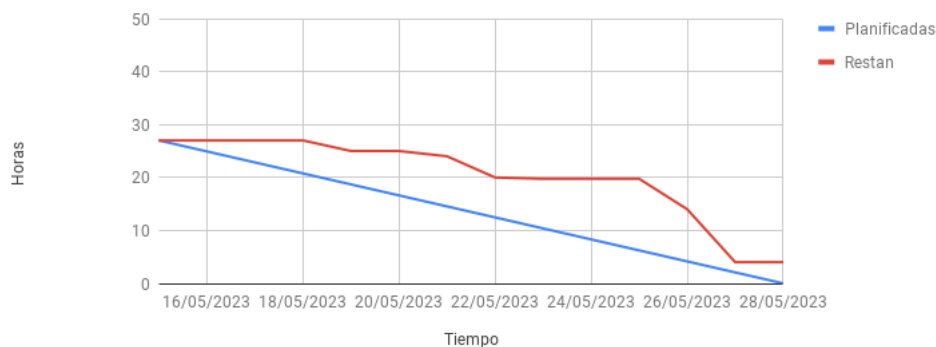


Figura 4.5: Gráfica de la progresión del Burndown en el Sprint 3

En esta iteración se cumplieron la mayoría de los objetivos que se propusieran. La funcionalidad básica de la app Android estaba completa, pero la TRA-29 se había quedado a medias de la implementación. Era un retraso bastante mínimo así que se decidió pasar lo que restaba de la tarea al siguiente sprint de nuevo.

4.7. Sprint 4

Esta iteración se implementarían las funcionalidades de cuentas de usuario e inicio de sesión, y para ello habría que desarrollar un backend al que pudieran conectarse tanto el editor web como la app y hacer las llamadas que necesitaran a la API para operaciones como crear un usuario o actualizar una escena. Sin embargo, a lo largo de la formación que se realizó en estas tecnologías y antes de redactar las tareas de la iteración, se llegó a la conclusión que se necesitaba más tiempo de desarrollo para implementarlo que lo que se había planeado para este sprint. Sobre todo ocurría que con todo el código que había que preparar tanto en el backend como en el frontend para conectar ambos, a penas daba tiempo a implementar correctamente las interfaces de usuario que se requerían. Además había que recuperar un pequeño retraso del sprint anterior.

Esto no supuso ningún golpe para el proyecto, ya que se anticipó una posible situación similar. El area en el que menos experiencia se tenía a la hora de comenzar el desarrollo era en el desarrollo de backend. Y aunque se hicieron estimaciones, se contemplaba la posibilidad de que pudieran ser erróneas desde

un inicio. Es por ello que para el sprint 5 se designaron las historias de usuario de menor prioridad de las cuales podía prescindir el proyecto si se diera el caso como la creación de figuras básicas, la filtración de escenas o las múltiples imágenes activadoras en imágenes aumentadas. Por tanto se tomó la decisión de descartar esas historias y reorganizar el Sprint 5. En el 4 se realizaron las siguientes tareas.

Tarea: TRA-31	Horas estimadas: 1	Horas totales: 2	HU asociada RA-10
Crear proyecto base de Node.js con Express.			

Tarea: TRA-32	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10
Crear y configurar proyecto de Google Firebase, obteniendo las claves secretas necesarias e incluyéndolas en el proyecto de Node.js.			

Tarea: TRA-33	Horas estimadas: 1	Horas totales: -	HU asociada RA-10
Implementar función en Node.js para el inicio de sesión de un usuario.			

Tarea: TRA-34	Horas estimadas: 1	Horas totales: -	HU asociada RA-10
Implementar función en Node.js para la creación de una cuenta de usuario.			

Tarea: TRA-35	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.2
Implementar función en Node.js para la creación de una nueva escena.			

Tarea: TRA-36	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.2
Implementar función en Node.js para eliminar una escena a partir de su identificador.			

Tarea: TRA-37	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.2
Implementar función en Node.js para actualizar una escena ya existente.			

Tarea: TRA-38	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.2
Implementar función en Node.js para obtener todas las escenas de un usuario concreto.			

Tarea: TRA-39	Horas estimadas: 1	Horas totales: 2	HU asociada RA-10.2
Configurar Firebase Storage para poder almacenar objetos binarios como imágenes o modelos 3D y que sean accesibles desde una url para descargarlos.			

Tarea: TRA-40	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.2
Diseñar las tablas de bases de datos para usuarios y escenas.			

Tarea: TRA-41	Horas estimadas: 2	Horas totales: 2	HU asociada RA-10.2
Implementar las tablas de bases de datos en Google Firebase.			

Tarea: TRA-42	Horas estimadas: 2	Horas totales: 2	HU asociada RA-12
Configurar acceso a Firebase Storage desde la app Android.			

Tarea: TRA-43	Horas estimadas: 10	Horas totales: 8	HU asociada RA-10
Configurar acceso a Firebase Storage y API desde React. Asignar a cada menú de la aplicación su llamada correspondiente.			

Tarea: TRA-44	Horas estimadas: 1	Horas totales: -	HU asociada RA-10
Implementar función en Node.js para la modificación de datos de un usuario.			

Tarea: TRA-45	Horas estimadas: 1	Horas totales: -	HU asociada RA-10
Implementar función en Node.js para la eliminación de un usuario.			

Algo importante a destacar en este Sprint es que las tareas TRA-33, TRA-34, TRA-44 y TRA-45 fueron descartadas durante el desarrollo. Esto se debe a que, como se comentará en el capítulo de Implementación, se decidió optar por otra vía en cuanto a la gestión de cuentas de usuario se refiere, por lo que ya no sería necesario crear estas llamadas en la API. En esta iteración se dispuso de más tiempo, lo cual se reflejó en la compleción de todas las tareas a tiempo.

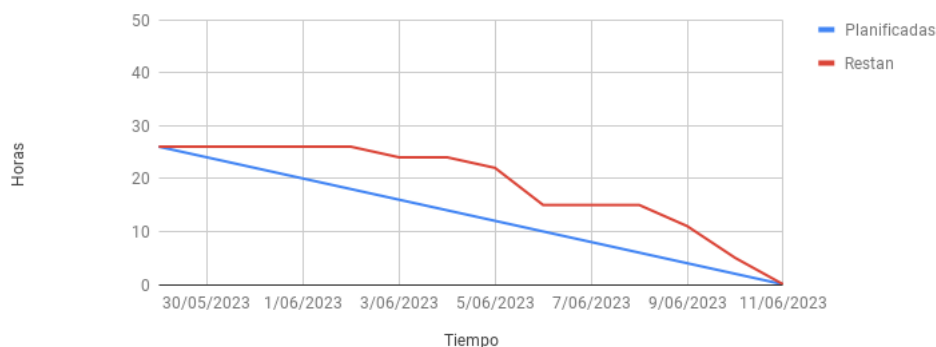


Figura 4.6: Gráfica de la progresión del Burndown en el Sprint 4

4.8. Sprint 5

Este sprint en un inicio se planificó como un colchón en caso de que la planificación no fuera al ritmo esperado en el resto de sprints. Se desarrollarían las historias RA-9, RA-9.1, RA-12.2 y RA-3, de prioridad LOW, sin las cuales el sistema podía existir sin perder demasiado. Como ya se comentó en anteriores sprints, se descartó la implementación de todas ellas, y en su lugar se continuó con RA-10, RA-10.2 y RA-12 del cuarto sprint, que requerían de más tiempo además de RA-10.1 y RA-12.1 de prioridad MID que sí estaban programadas para esta iteración.

Tarea: TRA-47	Horas estimadas: 4	Horas totales: 4	HU asociada RA-12.1
Implementar menú de selección de escena en la aplicación Android.			

Tarea: TRA-48	Horas estimadas: 4	Horas totales: 4	HU asociada RA-10.1
Implementar menú de selección de escena la aplicación web.			

Tarea: TRA-49	Horas estimadas: 2	Horas totales: 4	HU asociada RA-12.1
Configurar la API de acceso al backend para que sea accesible desde otros dispositivos de la red.			

Tarea: TRA-50	Horas estimadas: 3	Horas totales: 3	HU asociada RA-10
Implementar funciones para el inicio de sesión y registro de un usuario a partir de un correo y contraseña en la aplicación web.			

Tarea: TRA-51	Horas estimadas: 2	Horas totales: 2	HU asociada RA-12
Implementar inicio de sesión con usuario y contraseña en la aplicación Android.			

Tarea: TRA-53	Horas estimadas: 2	Horas totales: 4	HU asociada RA-12.1
Implementar petición GET desde la app Android a la API para obtener las escenas creadas por un usuario.			

Tarea: TRA-55	Horas estimadas: 1	Horas totales: 1	HU asociada RA-10.1
Implementar menú para el cambio de credenciales de un usuario en el editor web.			

Tarea: TRA-56	Horas estimadas: 3	Horas totales: 3	HU asociada RA-10
Proteger las rutas que no sean la página de login del editor web para solo poder acceder a ellas cuando hay una sesión iniciada.			

Tarea: TRA-57	Horas estimadas: 10	Horas totales: 7	HU asociada RA-10.2
Implementar la carga, guardado y borrado de una escena a partir del JSON obtenido por la API con Firebase Storage en el editor web.			

Tarea: TRA-58	Horas estimadas: 2	Horas totales: 1.5	HU asociada RA-10.2
Añadir animación de carga para momentos en los que se esté esperando la respuesta a una petición o la carga de algún recurso desde el editor web.			

Tarea: TRA-59	Horas estimadas: 3	Horas totales: 2	HU asociada RA-10.2
Implementar menús pop-up de confirmación para las acciones de eliminar una escena, abandonar el editor y cerrar sesión desde el editor web.			

El sprint finalizó sin mayores complicaciones, teniendo el producto final listo. A estas alturas se notó la experiencia obtenida en el proceso de planificación, el cual fue preciso. Una vez terminado el producto solo restaba desplegar la aplicación.

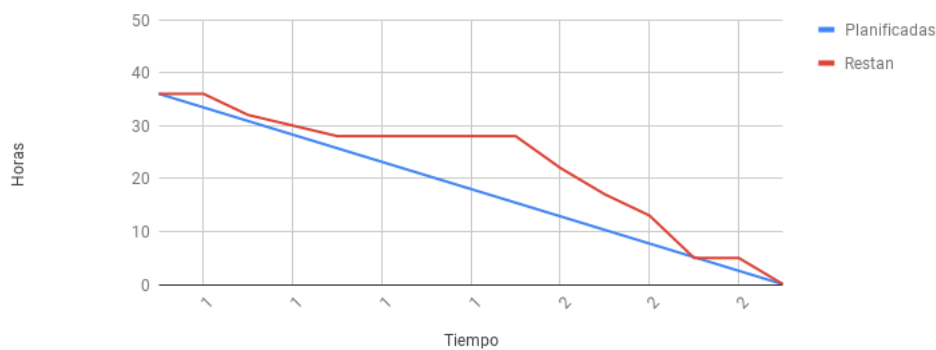


Figura 4.7: Gráfica de la progresión del Burndown en el Sprint 5

Capítulo 5

Implementación

La implementación del software se ha dividido en hitos. Estos, han sido definidos en Github y cada uno de ellos contiene un grupo de *issues* que se corresponden con las distintas mejoras que se han ido incorporando al software a lo largo de su desarrollo.

Capítulo 6

Conclusiones y trabajos futuros

Bibliografía

- [1] Pictarize studio. <https://pictarize.com/>.
- [2] Carmen Lasa Gómez Alonso Álvarez García, Rafael de las Heras del Dedo. Métodos Ágiles y SCRUM, manual imprescindible. .
- [3] Express.js. Fast, unopinionated, minimalist web framework for node.js. <http://expressjs.com/>.
- [4] Free Software Foundation. GNU General Public License. <http://www.gnu.org/licenses/gpl.html>.
- [5] Node.js Foundation. Node.js. <https://nodejs.org/es/>.
- [6] Google. Arcore. <https://developers.google.com/ar?hl=es-419>.
- [7] Google. Firebase. <https://firebase.google.com/?hl=es-419>.
- [8] Thomas Gorisse. Sceneview, 3d and ar view for android, flutter and react native working with arcore and google filament. <https://sceneview.github.io/>.
- [9] Javier González Jiménez. Visión por computador. .
- [10] Scrum Manager. Troncal i. scrum master. http://scrummanager.net/files/scrum_manager.pdf.