Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Desarrollo de una aplicación de realidad aumentada para entornos educativos

Autor:
Guillermo Muñoz Crespo

Tutores:
Alejandra Martínez Monés
Juan Ignacio Asensio Pérez

Resumen

El presente trabajo aborda el rediseño y actualización tecnológica de la aplicación *BusketServer* destinada a entornos educativos y cuyo propósito fundamental es dar soporte a los educadores en situaciones de aprendizaje ligadas a diferentes espacios, sean virtuales o físicos. En el primer caso, el entorno formativo se basa en desarrollos de tipo web mientras que en el segundo se hace uso del paradigma de la realidad aumentada. El objetivo último ha sido rediseñar y actualizar aquellos componentes e interfaces que debido al transcurso del tiempo se han quedado obsoletos facilitando de este modo al usuario la interacción con la aplicación.

Abstract

This project deals with the redesign and technological update of the BusketServer application for educational environments, whose main purpose is to support educators in learning situations linked to different spaces, whether virtual or physical. In the first case, the training environment is based on web-based developments, while in the second case, the augmented reality paradigm is used. The ultimate goal has been to redesign and update those components and interfaces that, due to the passage of time, have become obsolete, thus facilitating the user's interaction with the application.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

La realidad aumentada (AR, por sus siglas en inglés, *Augmented Reality*), es una tecnología que permite superponer información digital en objetos o lugares del mundo físico, para mejorar la experiencia del usuario. A diferencia de la realidad virtual (Virtual Reality, VR), que crea un entorno totalmente digital desarrollado por una computadora, la AR combina la realidad aumentada con la información digital [20].

La aplicación de *BucketServer* [24], desarrollada por investigadores de la Universidad de Valladolid, es un ejemplo de aplicación de la tecnología AR al ámbito educativo. Dicha aplicación se basa en el concepto de *learning buckets* [25]. Son contenedores digitales que pueden estar ubicados en una posición física o virtual y almacenan diversos tipos de artefactos de aprendizaje (*learning artifacts*), es decir, recursos que desempeñan una función específica. Estos recursos son creados tanto por el profesor como por los estudiantes y pueden ser compartidos entre actividades y espacios. La idea de los *learning buckets* es ayudar a los profesores en el aprendizaje de los alumnos tanto en espacios físicos como virtuales, gracias a la posibilidad de posicionar a los artefactos de aprendizaje de diferentes maneras (web, geoposición, marcadores). Su gran ventaja es la flexibilidad y el control que tienen tanto el profesor como el alumno. Los *buckets* se pueden configurar por los profesores para crear restricciones, dependiendo del uso que le queramos dar y crear un gran número de posibilidades. Por ejemplo hacerlo visible o invisible para determinados usuarios o seleccionar el número máximo de artefactos que pueda contener. Todo esto se puede hacer gracias a la aplicación *BucketServer*. Además permite diferentes formas de posicionamiento, para potenciar las actividades en diferentes espacios, no como ocurre en otras plataformas como *Moodle* donde se puede indicar el número máximo de ficheros, pero no el mínimo y de qué tipo tienes que entregar como alumno. Otra de las diferencias de *BucketServer* con *Moodle*, es que *Moodle* no esta conectado al espacio físico.

Los *learning artifacts* pueden ser de diferentes tipos: una foto, un *Google Document*, un enlace a *Wikipedia*... Los artefactos se pueden ubicar en diferentes espacios. Estos posicionamientos podrían ser en un código QR[1], a través de la aplicación *Google Maps* seleccionando las coordenadas, utilizando mundos virtuales en 3D como *Google Earth* o utilizando AR. Con todas esas tecnologías se permite el aprendizaje desde distintas perspectivas, permitiendo a los estudiantes aprender desde entornos dentro y fuera del aula. Un ejemplo sería el aprendizaje de la vegetación en un bosque conectado con una lección de biología en el aula.

[1]Quick Response Code (Código de Respuesta Rápida): Es un módulo para almacenar información que puede ser escaneada por un dispositivo móvil, para que te muestre una respuesta.

Dentro de la aplicación *BucketServer* se utilizan *Puntos de interés* (POIs, de sus siglas en inglés, *Points of Interest*). Los POIs son puntos de ubicación específica que alguien puede encontrar útil o interesante, para describir el espacio físico donde se van a ubicar los buckets y artefactos que se creen.

En la Figura **1.1** se muestra un ejemplo del funcionamiento de la aplicación *Bucketserver* con la plataforma *Moodle*. En tiempo de diseño (Design Time), un profesor crea y configura el *bucket* seleccionando el número máximo de artefactos que se pueden incluir en él, tipo de artefactos, tipo de posicionamiento... Más tarde, durante la clase presencial los alumnos pueden crear artefactos y posicionarlos como se muestra en la actividad 1 (*Activity* 1 en la Figura). Para la creación de artefactos la aplicación *BucketServer* se conecta con los proveedores de artefactos (Artifacts providers), que son los encargados de la creación de los artefactos y sus distintos tipos: *Google Document*, una foto, un enlace a *Wikipedia*... Una vez han sido creados, se puede acceder a ellos en diferentes espacios, con distintas aplicaciones de realidad aumentada, como pueden ser *Junaio*[2] o *Layar*[3]. El grupo 2 formado por alumnos como se muestra en la actividad 2 (*Activity* 2 en la Figura), acceden al posicionamiento de los artefactos con la aplicación *Junaio* para visualizarlos en realidad aumentada y poder disfrutar de la actividad interactiva y educativa. En la actividad 3 (*Activity* en la Figura), se trata de una alternativa a la actividad 2 donde el grupo 2 accede al posicionamiento de los artefactos con la aplicación *Google Earth*, para acceder a un mundo virtual en 3D.

Además de la aplicación *Junaio*, en la actividad 2, *BucketServer* utilizaba otras aplicaciones de realidad aumentada. Con el paso del tiempo éstas han quedado obsoletas y ha sido necesario buscar una alternativa tecnológica para poder suplir esa carencia dentro de la aplicación *BucketServer*.

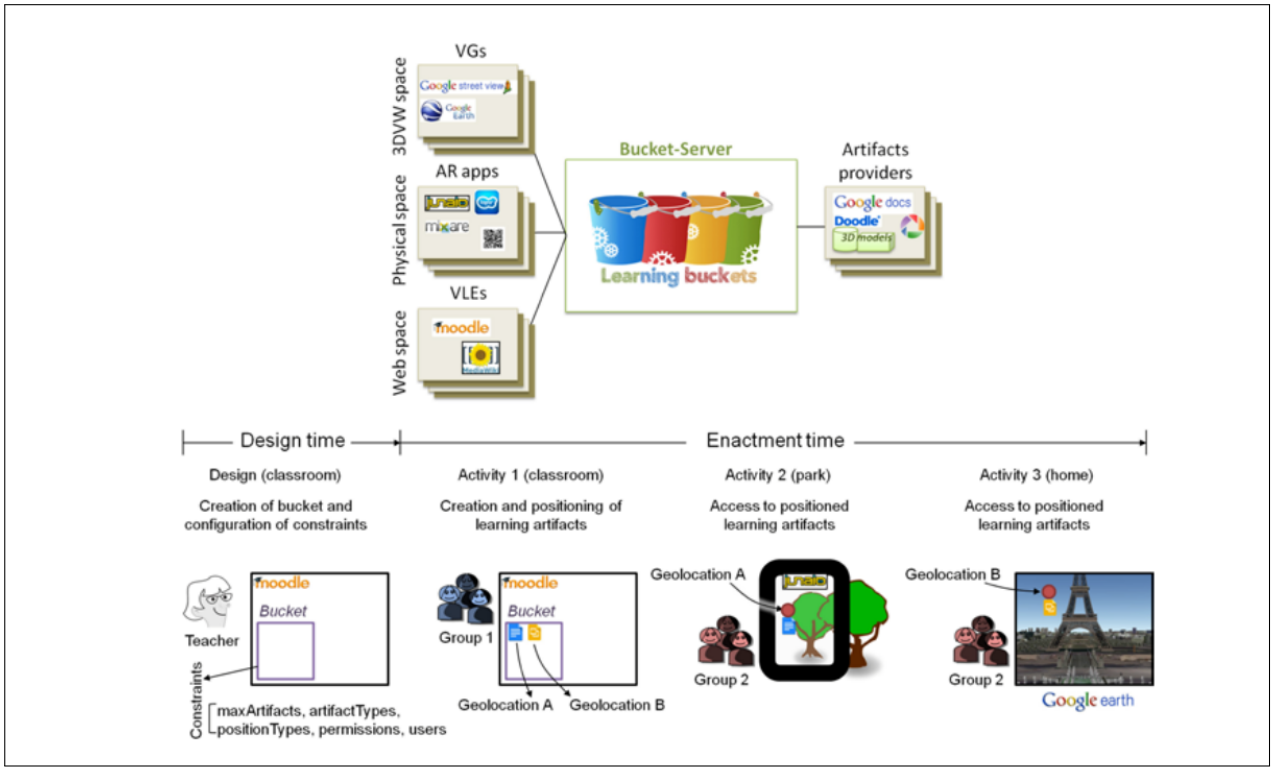


Figura 1.1: Sistema Bucket-Server y ejemplo de uso [24]

[2]Junaio - Era una aplicación móvil de un navegador AR para dispositivos móviles 3G y 4G: `https://en.wikipedia.org/wiki/Junaio`

[3]Navegador móvil que permite a los usuarios encontrar elementos en realidad aumentada: `https://www.layar.com/`

1.2. Motivación

Actualmente existe la necesidad de actualizar la parte de realidad aumentada dentro de la aplicación *BucketServer* debido a que las aplicaciones de realidad aumentada que utilizaba *BucketServer* han quedado obsoletas por diversos factores como: La venta de las aplicaciones a otras compañías y el cambio de funcionalidad de las aplicaciones.

BucketServer es una aplicación web educativa que se encarga de la creación y configuración de los *buckets* y artefactos, esta web almacena los datos en su base de datos. Estos datos eran utilizados en diversas actividades, como por ejemplo la visualización de un mundo virtual 3D o la visualización en realidad aumentada, en nuestro caso la actividad que más nos interesa es la de la realidad aumentada, que es en la que se centra la mayor parte del proyecto. La aplicación *BucketServer* se conectaba con las aplicaciones de realidad aumentada *Junaio*, *Layar*... a través de su API REST, creando un canal de comunicación entre dichas aplicaciones, enviándole los datos necesarios que habían sido almacenados con anterioridad en la base de datos de *BucketServer*, para poder crear puntos de interés dentro de la aplicación de realidad aumentada y poder visualizarlos. Además de la actualización de la parte de realidad aumentada dentro de la aplicación *BucketServer* era necesario actualizar la documentación, para poder ofrecer un mejor manejo de la aplicación y que el usuario fuera consciente de todas las funcionalidades que le puede brindar.

1.3. Alcance y objetivos

El objetivo del proyecto es renovar las partes en desuso de la aplicación *BucketServer*, enfocándose en la parte de AR de la aplicación.
Finalmente, los objetivos definidos son los siguientes:

- Seleccionar la tecnología AR más adecuada para suplir a las existentes.
- Actualizar la parte de realidad aumentada de la aplicación *BucketServer*:
  - Disponer en la aplicación *BucketServer* de una nueva funcionalidad de realidad aumentada, que permita reconocer marcadores para mostrar un artefacto de un *bucket* con la cámara del dispositivo.
  - Disponer en la aplicación *BucketServer* de una nueva funcionalidad de realidad aumentada, que permita visualizar los artefactos de un *bucket* con la cámara del dispositivo en su ubicación exacta.
- Actualizar la documentación para que la aplicación sea mantenible en un futuro.

1.4. Estructura del documento

La estructura que se ha utilizado en la memoria es la siguiente:

- **Capítulo 1. Introducción.** Presenta el TFG.
- **Capítulo 2. Análisis de tecnologías realidad aumentada.** Análisis comparativo de las aplicaciones AR, características y funciones.
- **Capítulo 3. Análisis de requisitos.** Define requisitos funcionales y no funcionales.

- **Capítulo 4. Plan de proyecto.** Organiza el proyecto, sus etapas y riesgos.
- **Capítulo 5. Modelo de análisis.** Describe casos de uso y diagramas de modelo de dominio.
- **Capítulo 6. Diseño.** Fase de diseño software.
- **Capítulo 7. Implementación.** Explica las técnicas y herramientas software utilizadas para el desarrollo del proyecto.
- **Capítulo 8. Plan de pruebas y validación.** Presenta las diferentes pruebas ejecutadas y su validación para la aplicación.
- **Capítulo 9. Conclusiones y trabajo futuro.** Describe la conclusión final, objetivos conseguidos, trabajo futuro y valoración personal.
- **Anexo I. Manual de instalación.** Describe la instalación de la aplicación.
- **Anexo II. Manual de usuario.** Explica el uso de la aplicación para el usuario.

Capítulo 2

Análisis de tecnologías de realidad aumentada

En este capítulo vamos a realizar un análisis de tecnologías AR, con el fin de seleccionar la más apropiada para las necesidades del proyecto.

2.1. Criterios de selección

Para el análisis de una tecnología es necesario utilizar una serie de criterios, para escoger la mejor herramienta AR:

- **App móvil AR/Bibliotecas AR web** - Definir si se trata de una aplicación móvil AR o bibliotecas AR, que permitan agregar nuevas funcionalidades dentro de la página web. Es necesaria una aplicación externa que se pueda vincular a *BucketServer* o utilizar bibliotecas que nos permitan introducir la funcionalidad de realidad aumentada.
- **Comunidad Activa** - La existencia de una comunidad que tenga una implicación en el proyecto, dándole apoyo y soporte, es muy importante, lo que supone la tecnología AR no quede en desuso. Además si va a existir un mantenimiento de la aplicación, es necesario un soporte por si ocurre algún problema.
- **Lenguajes de Programación** - Este criterio indica qué lenguaje de programación utiliza la tecnología que estamos analizando. Es importante, porque en igualdad de condiciones, se puede preferir una tecnología compatible con los lenguajes de programación utilizados en la aplicación existente como *Javascript*, *Java*...
- **API Web** - Disponibilidad de una API (Application Programming Interfaces) web bien documentada, en la que la aplicación móvil hace uso de un backend y se pueden realizar peticiones REST a esta API para obtener esos datos. Este criterio es fundamental porque sin la existencia de una API web no se pueden obtener los datos necesarios para el desarrollo de la funcionalidad de realidad aumentada.
- **Coste** - Especificar si la aplicación es de uso gratuito o si es de pago. Este criterio es importante porque siempre tendrá prioridad las tecnologías gratuitas a las de pago.
- **Tipo de posicionamiento** - Cada vez que se genere un artefacto y se quiera posicionar en un lugar de interés es necesario generar un POI con la herramienta AR que represente dicho artefacto. Este criterio es imprescindible para el uso de POIs en la aplicación *BucketServer*.

Dentro de su importancia, existen varias categorías de posicionamiento basándonos en los que ya tenían *BucketServer*:

- **Geoposición** - Permite crear puntos de interés, seleccionando las coordenadas GPS.
- **Marcador** - Permite utilizar marcadores para posicionar puntos de interés.
- **Imagen** - Utiliza una imagen como modo de posicionamiento para mostrar la información.

2.2. Listado de tecnologías seleccionadas

Inicialmente, vamos a realizar un estudio detallado de qué aplicaciones de AR hay disponibles y cuáles se adaptan mejor a nuestros criterios. En *Google Scholar*[1] y *Google*[2] fueron los motores de búsqueda, para encontrar información. Se encuentraron varias páginas de las mejores *apps* AR para el desarrollo *software* [18] [17]. También hice un estudio de las tecnologías que pudiesen añadir la parte de AR, dentro de una aplicación web.

Los resultados de las tecnologías encontradas y que vamos a analizar son:

- Wikitude [7]
- Vuforia [6]
- ARCore [1]
- EasyAR [5]
- MaxST [15]
- ARToolkit [4] [9]
- Kudan [13] [14]
- ARpoise [12] [3]
- GeoAR [8]
- Layar [10]
- AR.js [12]

2.3. Tabla de comparación de resultados

En esta sección hemos creado una tabla para la comparación de los resultados, para que se puedan observar las diferencias que hay de unas a otras tecnologías de forma clara, precisa y sencilla. Se puede observar en la Figura **2.1**

Observando la Figura **2.1** podemos deducir, que muchas de las tecnologías que hay en el mercado o bien no se ajustan a nuestros criterios o se han quedado obsoletas.

[1]Google Scholar - `https://scholar.google.es/schhp?hl=es`
[2]Google - `https://www.google.es/`

	Wikitude	Vuforia	ARCore	EasyAR	MaxST	ARToolkit	Kudan	ARPoise	GeoAR	Layar	AR.js
APP AR/Librerías AR web	App RA	App RA	App RA	App RA	App RA	App RA	App RA	App RA	App RA	App RA	Librerías AR web
Comunidad Activa	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
Lenguajes de Programación	Android, IOs, Unity, JavaScript	Android, Unity, IOs, Java, C++	Android, IOs, Unreal, Unity	Java, C, Kotlin, C++, Objective-c, swift	Android, Unity, iOs	Unity, Java, C	Objective-c, Java, Unity, Ios, Android	Unity	C#	Android Studio	JavaScript
API WEB	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
Tipo de posicionamiento	✗	✗	Geoposición	✗	Marcador	Marcador	Marcador	Geoposición	Geoposición	Geoposición	Geoposición, Marcadores e imagen
Coste	De pago	Gratuito pero muchas funciones son de pago	Gratuito	Gratuito pero muchas funciones son de pago	Gratuito pero algunas funciones son de pago	Gratuito	Gratuito.	Gratuito	Gratuito	Gratuito	Gratuito
Aplicación desde cero	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗

Figura 2.1: Tabla de comparación de las tecnologías AR

Dentro de las aplicaciones AR, *GeoAR* y *Layar* no tienen una comunidad activa, por lo que quedan descartadas. El criterio del tipo de posicionamiento, se cumple en la mayoría de aplicaciones que se crean de cero, a excepción de *MaxST* y *ARPoise*.

MaxST, carece de una API web cualificada para este fin, por lo tanto dentro de las aplicaciones AR la mejor opción es *ARPoise*.

Dentro de las bibliotecas AR web se encuentra AR.js. Estas tienen una comunidad activa, su lenguaje de programación es *JavaScript*, al igual que la aplicación *BucketServer*. Tiene una API clara y detallada de su funcionamiento. Pudiendo crear tres tipos de posicionamiento, lo que permite aumentar el número de posibilidades de las bibliotecas, siendo su uso es gratuito.

Las dos opciones posibles de uso, son la aplicación externa *ARPoise* o las bibliotecas de *AR.js*. *ARPoise* nos permite mantener la estructura del proyecto *BucketServer* de la parte de realidad aumentada, de la que se encargaba una aplicación externa. *AR.js* se puede utilizar dentro de la propia aplicación *BucketServer* y su lenguaje de programación es compatible, además permite crear tres tipos de posicionamiento, geoposición, marcador e imagen, mientras que *ARPoise* solo permite el tipo de posicionamiento de geoposición.

La decisión final ha sido utilizar la tecnología de bibliotecas AR de *AR.js*; con tres tipos de POIs ofrece un mayor número de posibilidades, es compatible con el lenguaje de programación que ya existía y su implementación se puede hacer de forma sencilla ya que requiere de pocas lineas de código para disponer de una pequeña demostración de cómo funciona.

2.4. AR.js

AR.js es una biblioteca para realidad aumentada en web, que tiene unas características de posicionamiento de imágenes, ubicación y marcadores [2]. Los puntos claves de esta biblioteca web son:

- **Basado en web**: Es una solución web, que no requiere de ninguna instalación. Utiliza *Javascript* basado en *three.js* + *A-Frame* + *jsartoolkit5*. Ar.js utiliza *jsartoolkit5* para el seguimiento, pero puede mostrar contenido aumentado con *three.js* o *A-Frame*.
- **Código abierto**: Es completamente gratuito y de código abierto.
- **Requisitos**: Funciona en cualquier teléfono con *webgl*[3] y *webrtc*[4].

Con esta tecnología se pueden crear tres tipos de posicionamiento de realidad aumentada: posicionamiento en una imagen, posicionamiento en unas coordenadas GPS o posicionamiento en un marcador.

[3]Método de generar gráficos 3D usando *JavaScript*, acelerado a través del *hardware*: `https://caniuse.com/webgl`
[4]Método para acceder a los datos de dispotivos externos como el vídeo de una *webcam*: `https://caniuse.com/stream`

2.4.1. Posicionamiento GPS

Para este caso es necesario solamente crear un archivo *html* donde se van a importar las bibliotecas de *AR.js* y definir el punto de las coordenadas GPS donde aparecerá el objeto. La etiqueta "a-scene" sirve para definir la escena de la cámara donde se definen los elementos que van a aparecer en la cámara. La etiqueta "a-text" es el texto que se va a mostrar, la cual tiene varios atributos: el valor que es el contenido del texto, *look-at* utiliza el GPS de la cámara para ubicarse y comparar las coordenadas, su escala y *gps-entity-place* son las coordenadas donde va a estar situado el texto.

```
    <!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>GeoAR.js demo</title>
    <script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
    <script src="https://unpkg.com/aframe-look-at-component@0.8.0/
    dist/aframe-look-at-component.min.js"></script>
    <script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/
    aframe-ar-nft.js"></script>
  </head>

  <body style="margin: 0; overflow: hidden;">
    <a-scene
      vr-mode-ui="enabled: false"
      embedded
      arjs="sourceType: webcam; debugUIEnabled: false;"
    >
      <a-text
        value="This content will always face you."
        look-at="[gps-camera]"
        scale="120 120 120"
        gps-entity-place="latitude: <add-your-latitude>; longitude: <add-your-longitude>;"
      ></a-text>
      <a-camera gps-camera rotation-reader> </a-camera>
    </a-scene>
  </body>
</html>
```

Dentro del dispositivo móvil solo es necesario activar el GPS móvil y navegar hasta la URL. Solo es necesario mirar alrededor y aparece un texto en la ubicación GPS solicitada.

2.4.2. Posicionamiento marcadores

Para este caso es necesario solamente crear un archivo *html* donde se van a importar las bibliotecas de *AR.js* y definir el tipo de marcador que se quiere usar para escanear. La etiqueta "a-scene" sirve para definir la escena de la cámara donde se definen los elementos que van a aparecer en la cámara. La etiqueta "a-marker" es para definir el tipo de marcador que se va a posicionar, la etiqueta "a-entity" es la entidad que se va a mostrar, la cual tiene varios atributos: la posición donde va a aparecer el elemento, su escala y el modelo que va a representar el elemento de la entidad.

```
    <!DOCTYPE html>
<html>
  <script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
  <!-- we import arjs version without NFT but with marker + location based support -->
  <script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar.js">
  </script>
  <body style="margin : 0px; overflow: hidden;">
    <a-scene embedded arjs>
      <a-marker preset="hiro">
        <a-entity
          position="0 0 0"
          scale="0.05 0.05 0.05"
          gltf-model="https://arjs-cors-proxy.herokuapp.com/https://raw.githack.com/
          AR-js-org/AR.js/master/aframe/examples/image-tracking/nft/trex/scene.gltf"
        ></a-entity>
      </a-marker>
      <a-entity camera></a-entity>
    </a-scene>
  </body>
</html>
```

Una vez se entra en la *url* solo se necesita enfocar al marcador con la cámara del dispositivo y se podrá visualizar el objeto.

2.4.3. Posicionamiento imagen

Para este caso es necesario solamente crear un archivo *html* donde se van a importar las bibliotecas de *AR.js* y definir la imagen que se va usar. La etiqueta "a-scene" sirve para definir la escena de la cámara donde se definen los elementos que van a aparecer en la cámara, la etiqueta "a-nft" es para definir los atributos del marcador de la imagen cuyas propiedades son el tipo de marcador de imagen, la ruta de la imagen con la que se va a comparar, *smooth* es para definir si se activa el modo estabilizador de la cámara para estabilizar la imágen, *smoothCount* es para definir el número de matrices que se van a utilizar para el modo de estabilizador cuantas más matrices mas lento irá el movimiento de la cámara, *smoothTolerance* la distancia desde donde se va a reconocer la imagen utilizando el modo suave y por último *smoothThreshold* es el umbral del estabilizador de la cámara que se mantendrá constante a menos que haya suficientes matrices por encima de la tolerancia. La etiqueta "a-entity" es la entidad que se va a mostrar, la cual tiene varios atributos: la posición donde va a aparecer el elemento, su escala y el modelo que va a representar el elemento de la entidad.

```
    <!DOCTYPE html>
<html>
  <script src="https://cdn.jsdelivr.net/gh/aframevr/
  aframe@1c2407b26c61958baa93967b5412487cd94b290b/dist/aframe-master.min.js"></script>
  <script src="https://raw.githack.com/AR-js-org/AR.js/master/
  aframe/build/aframe-ar-nft.js"></script>

  <body style="margin : 0px; overflow: hidden;">
  <div class="arjs-loader">
    <div>Loading, please wait...</div>
  </div>
      <a-scene
        vr-mode-ui="enabled: false;"
        renderer="logarithmicDepthBuffer: true;"
        embedded
        arjs="trackingMethod: best; sourceType: webcam;debugUIEnabled: false;"
      >
      <a-nft
        type="nft"
        url="https://arjs-cors-proxy.herokuapp.com/https://raw.githack.com/
        AR-js-org/AR.js/master/aframe/examples/image-tracking/nft/trex/trex-image/trex"
        smooth="true"
        smoothCount="10"
        smoothTolerance=".01"
        smoothThreshold="5"
      >
      <a-entity
        gltf-model="https://arjs-cors-proxy.herokuapp.com/https://raw.githack.com/
        AR-js-org/AR.js/master/aframe/examples/image-tracking/nft/trex/scene.gltf"
        scale="5 5 5"
        position="50 150 0"
      >
      </a-entity>
```

```
    </a-nft>
    <a-entity camera></a-entity>
  </a-scene>
</body>
</html>
```

Una vez se entra en la *url* solo se necesita enfocar a la imagen con la cámara del dispositivo y se podrá visualizar el objeto.

Capítulo 3

Análisis de requisitos

Este capítulo está dedicado al análisis de requisitos. En él se ofrece una visión de los diferentes actores que interactúan con el sistema, así como los requisitos (funcionales, no funcionales y de información). El objetivo de este capítulo es proporcionar una base que permita realizar la planificación del proyecto, para después realizar la fase de análisis de la aplicación.

3.1. Definición de actores

A continuación, se describen los distintos tipos de actores que interaccionan con el sistema. Los actores interaccionan con el sistema, pero son externos al sistema y son los encargados de llevar a cabo un caso de uso. Un caso de uso siempre es iniciado por un actor, que le envía un mensaje o estímulo al sistema [23].

3.1.1. Administrador

El actor administrador es el encargado de gestionar los roles de usuario para definir quién es un *Alumno* y un *Profesor*. El actor administrador también puede ser un actor *Profesor*.

3.1.2. Profesor

El actor *Profesor* se encarga de crear, editar y eliminar *buckets*. Los profesores pueden determinar ciertas restricciones, definiendo el número máximo de artefactos que se pueden crear, su tipo, tipo de posicionamiento, qué *Alumno* puede acceder al *bucket*... El actor *Alumno* y *Profesor* comparten varias funcionalidades como se especifica en el actor *Alumno*.

3.1.3. Alumno

El actor *Alumno* se encarga de añadir artefactos dentro de un bucket definido por un *Profesor*. Dentro del *bucket* el *Alumno* o *Profesor* puede seleccionar el tipo, el nombre, su descripción o el tipo de posicionamiento, seleccionando una de las cuatro opciones siguientes:

- Ningún posicionamiento
- Geoposicionar
- Posicionar en un código QR
- Posicionar en un marcador barcode

Tanto *Alumno* como *Profesor* puede seleccionar entre distintos tipos de vista para la visualización de los artefactos dentro de un *bucket*:

- Vista que muestra el artefacto incrustado en la página web, pudiendo ser una foto, documento...
- Vista que muestra todos los marcadores de los artefactos, ya sean códigos QR o *barcodes*.
- Vista que expone la ubicación del artefacto en *Google Maps*.
- Vista que reconoce los marcadores *barcode*, para mostrar el artefacto en realidad aumentada
- Vista que reconoce la geoposición de un artefacto que lo muestra por proximidad en realidad aumentada.

3.1.4. Proveedor de artefactos

El actor *Proveedor de artefactos* proporciona los diferentes tipos de artefactos a la aplicación *BucketServer*. Cuando un actor *Profesor* o *Alumno* crea un artefacto, por ejemplo un *Google Document*, la aplicación *BucketServer* se conecta con el actor *Proveedor de artefactos* que es el encargado de crear este tipo de artefacto.

3.2. Requisitos funcionales

- RF-1: El sistema debe permitir al *Profesor* o *Alumno* seleccionar la geoposición de un artefacto escribiendo las coordenadas de forma manual o usando el mapa de Google Maps.
- RF-2: El sistema debe permitir al *Profesor* o *Alumno* seleccionar el tipo de posicionamiento con marcador para el artefacto.
- RF-3: El sistema debe permitir al *Profesor* o *Alumno* ver las coordenadas GPS relacionadas con el artefacto.
- RF-4: El sistema debe permitir al *Profesor* o *Alumnos* modificar las coordenadas GPS vinculadas al artefacto por otras.
- RF-5: El sistema debe permitir al *Profesor* o *Alumnos* modificar el marcador de un artefacto por otro.
- RF-6: El sistema debe permitir al *Profesor* o *Alumno* crear un artefacto, comunicándose con el actor *Proveedor de artefactos*.
- RF-7: El sistema debe permitir al *Profesor* o *Alumno* editar un artefacto, comunicándose con el actor *Proveedor de artefactos*.
- RF-8: El sistema debe permitir al *Profesor* o *Alumno* eliminar un artefacto, comunicándose con el actor *Proveedor de artefactos*.
- RF-9: El sistema debe permitir al *Profesor* o *Alumno* puede seleccionar un tipo de vista para ver todos los artefactos y su contenido.
- RF-10: El sistema debe permitir al *Profesor* o *Alumno* seleccionar un tipo de vista para ver los marcadores y códigos QR de los artefactos.
- RF-11: El sistema debe permitir al *Profesor* o *Alumno* seleccionar un tipo de vista que muestre Google Maps con los artefactos.

- RF-12: El sistema debe permitir al *Profesor* o *Alumno* acceder al modo de realidad aumentada, siempre y cuando existan artefactos con marcadores o con coordenadas GPS.
- RF-13: El sistema debe permitir al *Profesor* o *Alumno* cambiar dentro del modo de realidad aumentada entre visualizar artefactos con marcadores en AR o visualizar artefactos basados en su ubicación en AR.
- RF-14: El sistema cuando detecta un artefacto dentro del modo de AR, debe mostrar al *Profesor* o *Alumno* un modelo representativo del artefacto en realidad aumentada.
- RF-15: El sistema debe permitir dentro del modo de AR al *Profesor* o *Alumno* acceder al *bucket*.
- RF-16: El sistema debe permitir al *Profesor* o *Alumno* acceder a un artefacto.
- RF-17: El sistema debe permitir al *Profesor* crear un *bucket*.
- RF-18: El sistema debe permitir al *Profesor* editar un *bucket*.
- RF-19: El sistema debe permitir al *Profesor* eliminar un *bucket*.
- RF-20: El sistema debe permitir al *Administrador* dar de alta a un *Alumno* o *Profesor*.
- RF-21: El sistema debe permitir al *Administrador* dar de baja a un *Alumno* o *Profesor*.
- RF-22: El sistema debe permitir al *Administrador* editar un usuario *Alumno* o *Profesor*.
- RF-23: El sistema debe permitir identificar el usuario que accede.
- RF-24: El sistema debe mostrar al *Profesor* los *buckets* que ha creado.
- RF-25: El sistema debe mostrar al *Alumno* todos los *buckets* en los que un *Profesor* les ha dado acceso.
- RF-26: El sistema debe mostrar al *Profesor* o *Alumno* todos los *artefactos* que existen dentro de un *bucket*.

3.3. Requisitos no funcionales

- RNF-1: La aplicación debe tener compatibilidad con todas las tecnologías ya existentes
- RNF-2: La aplicación debe permitir al *Profesor* o *Alumno* realizar todas las operaciones con una mano desde el dispositivo móvil.
- RNF-3: El sistema deberá de estar disponible 24 horas, los 365 días del año.
- RNF-4: La tasa de errores por un Usuario debe de ser menor del 5 % para poder crear un *bucket* o artefacto.
- RNF-5: El sistema codificará la uri por motivos de seguridad, para que el contenido no sea legible por los usuarios.

3.4. Requisitos de información

Enumeración de los datos necesarios para el correcto funcionamiento de la realidad aumentada teniendo en cuenta la documentación de *AR.js*.

- Para la parte de realidad aumentada de los artefactos que tienen posicionamiento de geoposición es necesario:

  1. Identificador de cada artefacto.
  2. Nombre del artefacto para que se muestre en pantalla en la ubicación correspondiente.
  3. URL del *bucket* para acceder al *bucket*.
  4. Latitud.
  5. Longitud.

- Para la parte de realidad aumentada de los artefactos que tienen posicionamiento de marcadores es necesario:

  1. Identificador de cada artefacto.
  2. Nombre del artefacto para que se muestre en pantalla cuando se reconoce el marcador del artefacto.
  3. URL del artefacto para acceder al artefacto si se reconoce el marcador.
  4. URL del *bucket* para acceder al *bucket*.
  5. Valor del marcador *barcode*.

Capítulo 4

Plan de proyecto

4.1. Planificación

En esta sección se detalla tanto el plan de desarrollo del proyecto *software*, de la aplicación objetivo del TFG.

4.1.1. Ciclo de vida del proyecto

El ciclo de vida de este proyecto se divide en dos partes. En la primera parte del trabajo se ha tenido que realizar un trabajo de ingeniería inversa, para entender la aplicación *BucketServer*, debido a que la documentación de la aplicación era escasa y conseguir que funcionase en nuestro dispositivo. En la segunda parte ha sido el desarrollo de las funcionalidades de AR que se habían quedado desfasadas dentro de *BucketServer*.

En este proyecto se han utilizado algunas técnicas de ingeniería inversa [21] para conseguir los siguientes objetivos:

- La actualización de la información de la documentación de la aplicación *BucketServer*, ha consistido en crear una guía detallada sobre la instalación de la aplicación.
- Comprender el funcionamiento interno de la aplicación *BucketServer*, ha sido necesario para poder implementar la nueva parte de realidad aumentada dentro de la aplicación.

Para la segunda parte del proyecto del desarrollo *software* de AR, se ha seguido una metodología en cascada. Esta metodología ha ido evolucionando. Actualmente el modelo en cascada presenta el desarrollo software como una serie de etapas de forma lineal, pasando de una etapa a la siguiente.

Las etapas de este modelo son las siguientes [26], como se puede apreciar en la Figura **4.1**:

1. Análisis y definición de los requisitos. Se establecen los servicios, las limitaciones y los objetivos del sistema.
2. Diseño de sistemas y *software*. Asigna los requisitos al *hardware* o al *software*. Establece la arquitectura del sistema.
3. Implementación y pruebas unitarias. Durante esta etapa, el diseño software se realiza como un conjunto de programas o unidades de programa.
4. Integración y pruebas del sistema. Los programas se integran y se prueban como un sistema completo para garantizar que se cumplan los requisitos. Después de que se pruebe se entrega al cliente.

5. Operación y mantenimiento. El sistema se instala y se pone en práctica. El mantenimiento implica corregir los errores que no se descubrieron, mejorar la implementación de las unidades del sistema en conjunto con los servicios del sistema a medida que se descubren nuevos requisitos.

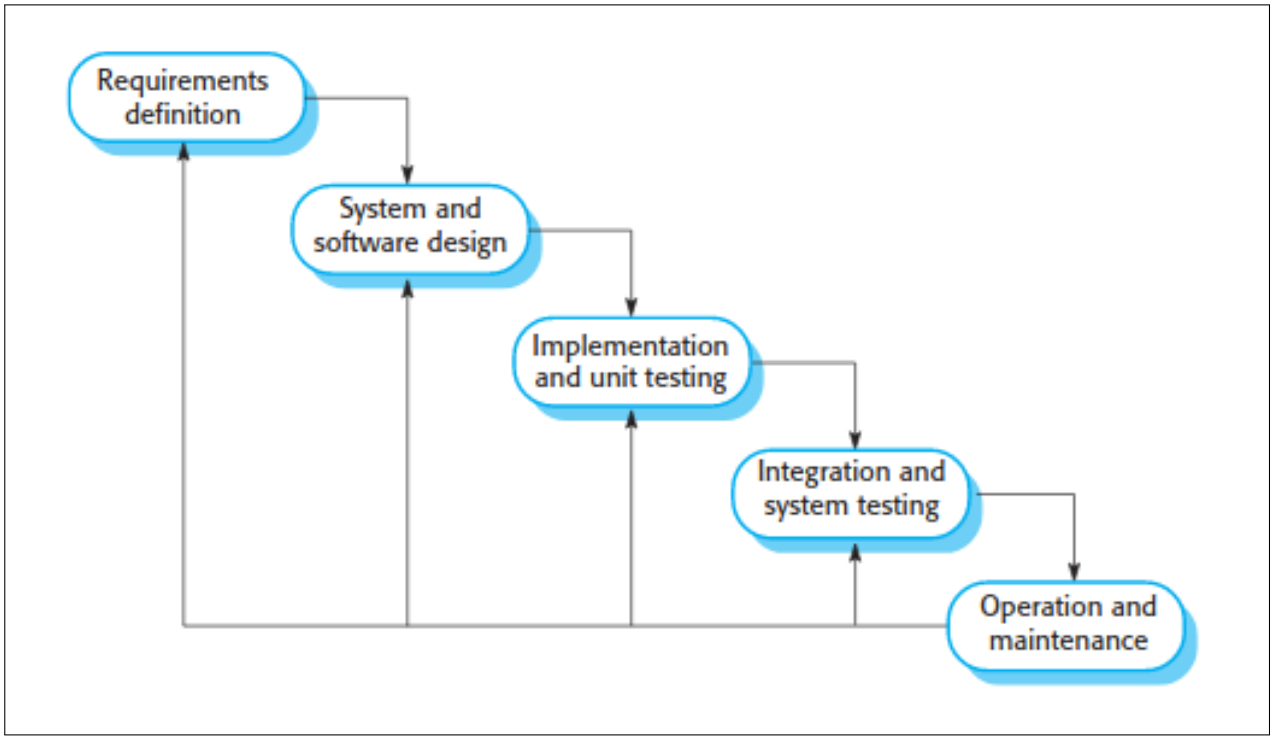


Figura 4.1: Modelo en cascada [26]

El desarrollo de este proyecto no es un desarrollo a largo plazo que conlleve muchas horas, los requisitos están claros y bien definidos y no van a cambiar a lo largo del proceso. Además las etapas del modelo en cascada se ajusta, a las necesidades del desarrollo, porque tenemos que realizar un análisis de las tecnologías AR en el mercado, estudiar los requisitos del software, crear un diseño software de la arquitectura del sistema, implementación del sistema programando el software, búsqueda de errores y pruebas unitarias, pruebas del sistema de su correcto funcionamiento y por último la etapa en la que se entrega la aplicación.

4.2. Gestión del proceso

En este punto se especifican los distintos planes para la gestión del proceso.

4.2.1. Plan inicial

La planificación de este proyecto se realizará por estimación, teniendo en cuenta como referencia la propia experiencia y otros de características similares. Dado que el proyecto se realiza de forma individual, no es necesario coordinarse con otros grupos de trabajo.

Para la realización son necesarias las siguientes habilidades:

- Conocimientos de JavaScript, HTML, CSS y Java.
- Conocimientos de MySQL y de lenguaje SQL.

- Conocimiento sobre despliegue de aplicaciones y puesta en marcha de servidores Apache.
- Conocimientos sobre planificación de proyectos.

Para este TFG disponemos de los recursos del grupo de investigación GSIC/EMIC.

4.2.2. Plan de trabajo

En este punto vamos a detallar las diferentes actividades en las que se divide el proyecto, así como el cómputo de horas del mismo. También tenemos que tener en cuenta que para el desarrollo de este proyecto se realizó la búsqueda de tecnologías AR, la actualización de la documentación sobre la aplicación ya existente del *BucketServer* y el despliegue de esta aplicación en un dispositivo personal, para poder trabajar luego con la aplicación para el desarrollo software.

Resumen total del proyecto

ID: -	**Resumen total del proyecto**
Estimado:	276 horas
Duración:	313 horas

Fase de análisis

ID: -	**Resumen de la Fase de Análisis**
Estimado:	73 horas
Duración:	95 horas

ID: 01	**Inicio de la Fase de Análisis**
Predecesoras:	-
Estimado:	-
Duración:	-
-	

ID: 02	**Reconocimiento de la aplicación *BucketServer***
Predecesoras:	01
Estimado:	5 horas
Duración:	7 horas
Esta actividad consiste en el reconocimiento de las tecnologías utilizadas y familiarizarnos con el código ya creado de la aplicación *BucketServer*.	

ID: 03	**Obtención de requisitos para las tecnologías AR**
Predecesoras:	02
Estimado:	10 horas
Duración:	8 horas
Obtención de requisitos necesarios para la tecnología AR y su acoplamiento con la aplicación *BucketServer*	

ID: 04	Listado de tecnologías AR disponibles
Predecesoras:	03
Estimado:	28 horas
Duración:	56 horas
Crear un listado de tecnologías AR con sus pros y contras, para poder seleccionar la más idónea	

ID: 05	Validación de tecnologías AR
Predecesoras:	04
Estimado:	20 horas
Duración:	15 horas
Realizar pequeños test a las tecnologías de como funcionan y finalmente cuáles pueden ser de utilidad para la aplicación *BucketServer*	

ID: 06	Planificación de riesgos
Predecesoras:	05
Estimado:	8 horas
Duración:	7 horas
Se estudiarán los riesgos del proyecto, cuáles podrían ser sus impactos y su plan de gestión	

ID: 07	Control de versiones
Predecesoras:	06
Estimado:	2 horas
Duración:	2 horas
Se utilizará un control de versiones para el proyecto y así evitar pérdidas de datos	

ID: 07	Fin de la fase de análisis
Predecesoras:	07
Estimado:	-
Duración:	-
-	

Fase de diseño

ID: -	Resumen de la Fase de Diseño
Estimado:	72 horas
Duración:	59 horas

ID: 08	Inicio de la Fase de Diseño
Predecesoras:	06
Estimado:	-
Duración:	-
-	

ID: 09	Modelado de Casos de Uso
Predecesoras:	08
Estimado:	12 horas
Duración:	16 horas
Se redactarán los casos de uso a partir del documento de requisitos, obteniendo además el diagrama de casos de uso	

ID: 10	Modelo de dominio
Predecesoras:	09
Estimado:	8 horas
Duración:	2 horas
Se elaborará el modelo de dominio del proyecto con sus clases pertinentes	

ID: 11	Diseño de los diagramas de secuencia
Predecesoras:	10
Estimado:	24 horas
Duración:	21 horas
Se elaborará los diagramas de secuencia en función de los casos de uso que han sido detallados	

ID: 12	Diseño de la arquitectura del sistema
Predecesoras:	11
Estimado:	15 horas
Duración:	12 horas
A partir de la arquitectura del sistema ya existente de la aplicación *BucketServer* y con el modelo de dominio, se elaborará un nuevo diseño de la arquitectura del sistema	

ID: 13	Diseño de la interfaz de usuario
Predecesoras:	12
Estimado:	13 horas
Duración:	8 horas
Diseñar la interfaz de usuario de realidad aumentada para la aplicación *BucketServer*	

ID: 14	Fin de la fase de diseño
Predecesoras:	13
Estimado:	-
Duración:	-
-	

Fase de Implementación

ID: -	Resumen de la Fase de Implementación
Estimado:	116 horas
Duración:	144 horas

ID: 15	**Inicio de la Fase de Implementación**
Predecesoras:	14
Estimado:	-
Duración:	-
-	

ID: 16	**Configuración del entorno de desarrollo**
Predecesoras:	15
Estimado:	56 horas
Duración:	76 horas
En esta actividad se ha realizado la primera parte del ciclo de vida del proyecto. Utilizando ingeniería inversa en la aplicación *BucketServer*, se logró la instalación y configuración en un ordenador personal, junto a la instalación de un servidor *Apache* para utilizarlo durante el desarrollo.	

ID: 17	**Ampliación del backend**
Predecesoras:	16
Estimado:	20 horas
Duración:	14 horas
En esta actividad se han realizado cambios en el código que ya había, creando nuevas funciones.	

ID: 17.1	**Añadir una función *MarkerAR* para obtener artefactos con marcadores *barcode***
Predecesoras:	16
Estimado:	4 horas
Duración:	4 horas
Nueva función en los archivos *JavaScript*: que muestre solamente los artefactos que tengan marcadores *barcode*.	

ID: 17.2	**Añadir una función *MarkerARView***
Predecesoras:	16
Estimado:	8 horas
Duración:	6 horas
Nueva función en los archivos *JavaScript*: que obtenga los parámetros necesarios de los artefactos con marcadores *barcode* del *backend*.	

ID: 17.3	**Añadir una función *LocationARView***
Predecesoras:	16
Estimado:	8 horas
Duración:	4 horas
Nueva función en los archivos *JavaScript*: que obtenga los parámetros necesarios de los artefactos con posicionamiento de geoposición del *backend*.	

ID: 18	Desarrollo *frontend*
Predecesoras:	17, 17.1, 17.2, 17.3
Estimado:	40 horas
Duración:	54 horas
Se desarrollará la parte de la vista de AR de la aplicación *BucketServer* utilizando *CSS*, *JavaScript* y *HTML*.	

ID: 18.1	Desarrollo de la vista para los artefactos con posicionamiento basado en marcadores *barcode*
Predecesoras:	17, 17.1, 17.2, 17.3
Estimado:	20 horas
Duración:	24 horas
Desarrollo de una nueva vista para los artefactos con marcadores *barcode*, que muestren con la cámara del dispositivo el artefacto en realidad aumentada.	

ID: 18.2	Desarrollo de la vista para los artefactos con posicionamiento de geoposición
Predecesoras:	17, 17.1, 17.2, 17.3
Estimado:	20 horas
Duración:	30 horas
Desarrollo de una vista para los artefactos con posicionamiento de geoposición, que muestren con la cámara del dispositivo el artefacto en realidad aumentada.	

ID: 19	Fin de la fase de implementación
Predecesoras:	18, 18.1, 18.2
Estimado:	-
Duración:	-

Fase de tests del sistema

ID: -	Resumen de la fase de tests del sistema
Estimado:	15 horas
Duración:	15 horas

ID: 20	Inicio de la fase de tests del sistema
Predecesoras:	19
Duración:	-
Estimado:	-
-	

ID: 21	Pruebas finales de la aplicación *BucketServer*
Predecesoras:	20
Estimado:	4 horas
Duración:	5 horas
Fase de testeo para comprobar el correcto funcionamiento de la aplicación	

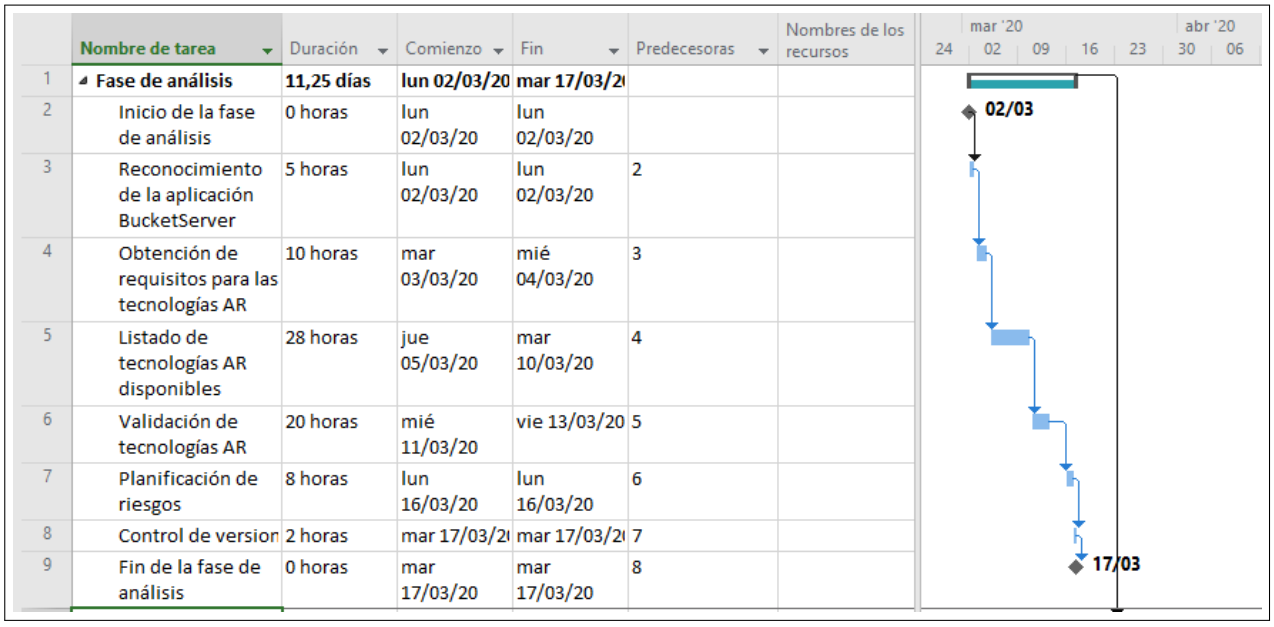
ID: 22	Elaboración de un manual de instalación
Predecesoras:	21
Estimado:	6 horas
Duración:	6 horas
Elaborar una guía para la instalación de la aplicación *BucketServer*	

ID: 23	Elaboración de un manual de usuario
Predecesoras:	22
Estimado:	5 horas
Duración:	4 horas
Elaborar una guía para el uso de la aplicación *BucketServer*	

ID: 24	Fin de la fase de tests del sistema
Predecesoras:	23
Estimado:	-
Duración:	-

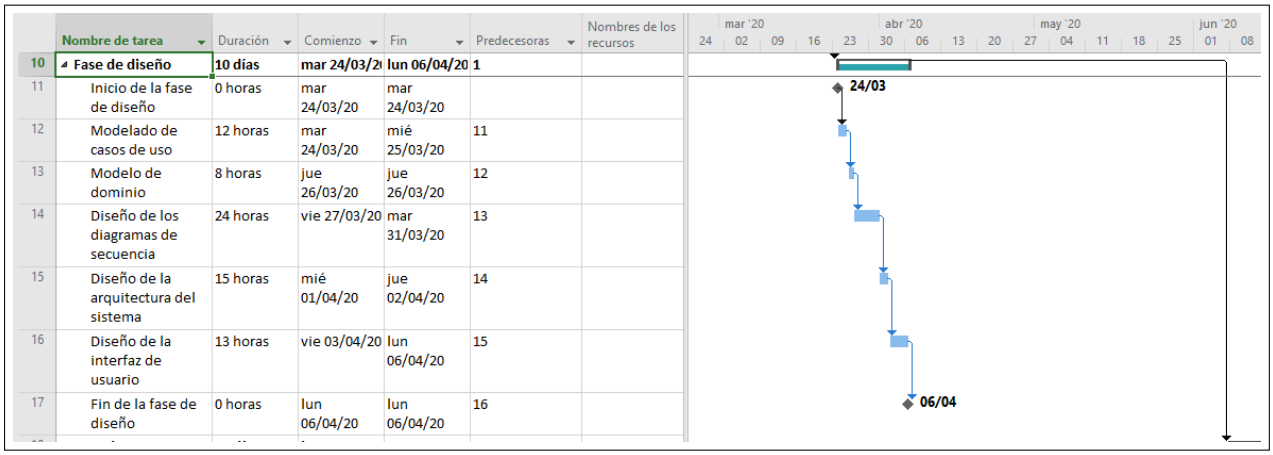
Diagramas de Gantt

A continuación, se añaden los diagramas de Gantt de las diferentes fases del proyecto.



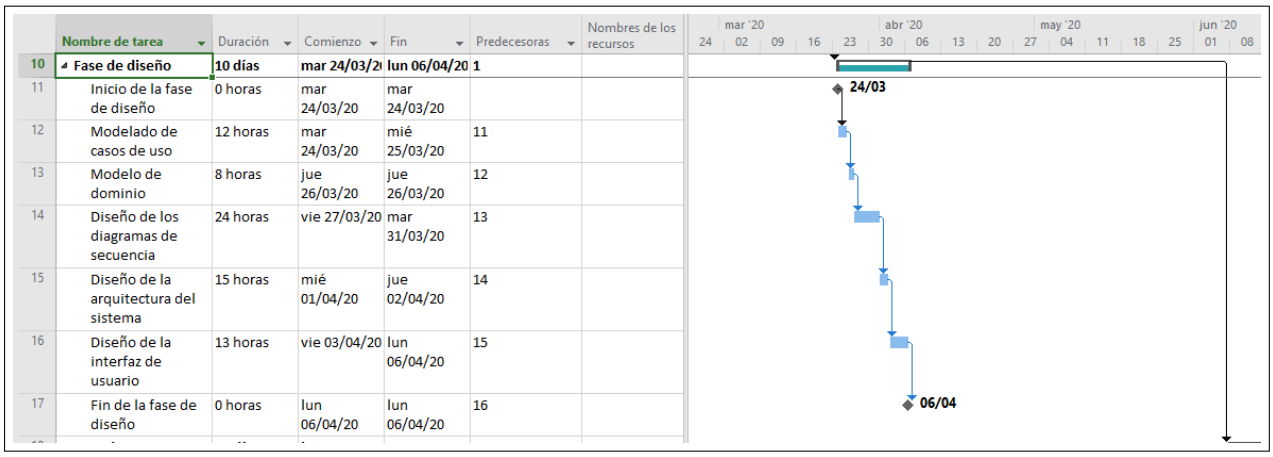
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Fase de análisis**	**11,25 días**	**lun 02/03/20**	**mar 17/03/2**		
2	Inicio de la fase de análisis	0 horas	lun 02/03/20	lun 02/03/20		
3	Reconocimiento de la aplicación BucketServer	5 horas	lun 02/03/20	lun 02/03/20	2	
4	Obtención de requisitos para las tecnologías AR	10 horas	mar 03/03/20	mié 04/03/20	3	
5	Listado de tecnologías AR disponibles	28 horas	jue 05/03/20	mar 10/03/20	4	
6	Validación de tecnologías AR	20 horas	mié 11/03/20	vie 13/03/20	5	
7	Planificación de riesgos	8 horas	lun 16/03/20	lun 16/03/20	6	
8	Control de version	2 horas	mar 17/03/2	mar 17/03/2	7	
9	Fin de la fase de análisis	0 horas	mar 17/03/20	mar 17/03/20	8	

Figura 4.2: Fase de análisis



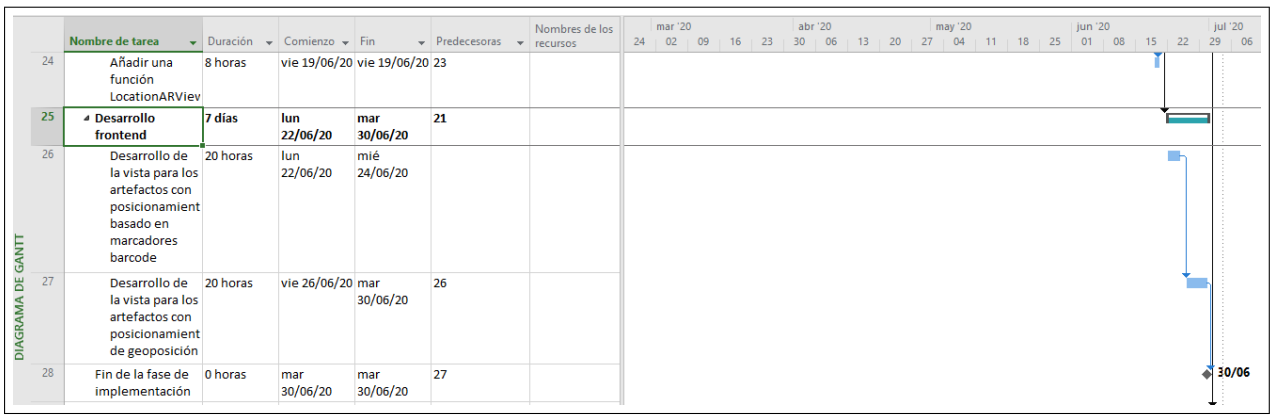
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
10	**Fase de diseño**	**10 días**	**mar 24/03/2**	**lun 06/04/20**	**1**	
11	Inicio de la fase de diseño	0 horas	mar 24/03/20	mar 24/03/20		
12	Modelado de casos de uso	12 horas	mar 24/03/20	mié 25/03/20	11	
13	Modelo de dominio	8 horas	jue 26/03/20	jue 26/03/20	12	
14	Diseño de los diagramas de secuencia	24 horas	vie 27/03/20	mar 31/03/20	13	
15	Diseño de la arquitectura del sistema	15 horas	mié 01/04/20	jue 02/04/20	14	
16	Diseño de la interfaz de usuario	13 horas	vie 03/04/20	lun 06/04/20	15	
17	Fin de la fase de diseño	0 horas	lun 06/04/20	lun 06/04/20	16	

Figura 4.3: Fase de diseño



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
10	**Fase de diseño**	**10 días**	**mar 24/03/2**	**lun 06/04/20**	**1**	
11	Inicio de la fase de diseño	0 horas	mar 24/03/20	mar 24/03/20		
12	Modelado de casos de uso	12 horas	mar 24/03/20	mié 25/03/20	11	
13	Modelo de dominio	8 horas	jue 26/03/20	jue 26/03/20	12	
14	Diseño de los diagramas de secuencia	24 horas	vie 27/03/20	mar 31/03/20	13	
15	Diseño de la arquitectura del sistema	15 horas	mié 01/04/20	jue 02/04/20	14	
16	Diseño de la interfaz de usuario	13 horas	vie 03/04/20	lun 06/04/20	15	
17	Fin de la fase de diseño	0 horas	lun 06/04/20	lun 06/04/20	16	

Figura 4.4: Fase de implementación 1º parte



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
24	Añadir una función LocationARViev	8 horas	vie 19/06/20	vie 19/06/20	23	
25	**Desarrollo frontend**	**7 días**	**lun 22/06/20**	**mar 30/06/20**	**21**	
26	Desarrollo de la vista para los artefactos con posicionamient basado en marcadores barcode	20 horas	lun 22/06/20	mié 24/06/20		
27	Desarrollo de la vista para los artefactos con posicionamient de geoposición	20 horas	vie 26/06/20	mar 30/06/20	26	
28	Fin de la fase de implementación	0 horas	mar 30/06/20	mar 30/06/20	27	

Figura 4.5: Fase de implementación 2º parte

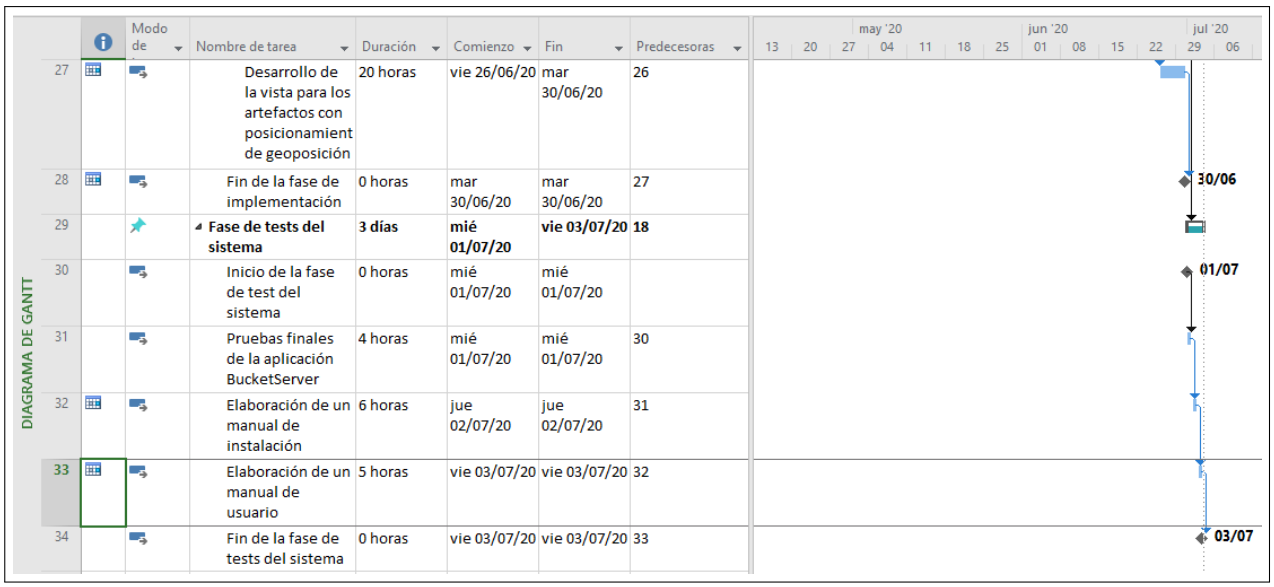


Figura 4.6: Fase de tests del sistema

4.3. Plan de gestión de riesgos

A continuación se exponen tanto los riesgos detectados, con una descripción de los mismos, su impacto y probabilidad, como la matriz de exposición mostrada en la tabla **4.1**. Con ambos datos se obtiene la exposición al riesgo.

Impacto\ Probabilidad	Muy Alto	Alto	Medio	Bajo	Muy Bajo
Catastrófico	Alto	Alto	Moderado	Moderado	Bajo
Crítico	Alto	Alto	Moderado	Bajo	Ninguno
Marginal	Moderado	Moderado	Bajo	Ninguno	Ninguno
Despreciable	Moderado	Bajo	Bajo	Ninguno	Ninguno

Tabla 4.1: Matriz Impacto/Probabilidad

4.3.1. Listado de riesgos

Realizamos un listado con los distintos riesgos que hemos podido observar, que podrían ocurrir durante el transcurso del proyecto.

R01 - Enfermedad

R01	**Enfermedad**
Descripción	No se puede trabajar según lo planificado debido a enfermedad.
Impacto	Marginal
Probabilidad	Alto
Exposición	Moderado
Plan de Protección	utilizar siempre mascarilla en espacios públicos, distancia de seguridad mínima de 1,5 metros, lavarse con frecuencia las manos y sino es así la utilización de geles hidroalcóholico
Plan de Contingencia	Evaluar el tiempo perdido y replanificar la actividad del proyecto.

Tabla 4.2: Riesgo 01 - Enfermedad

R02 - Pérdida de datos

R02	**Pérdida de datos**
Descripción	Pérdida de documentos o código con la consecuente pérdida del trabajo y tiempo invertidos.
Impacto	Crítico
Probabilidad	Baja
Exposición	Baja
Plan de Protección	Utilizar constantemente el control de versiones en un servidor principal y uno de respaldo para mantener la evolución de los documentos o código en lugar seguro.
Plan de Contingencia	En caso de pérdida de datos, habrá que evaluar el impacto sobre el proyecto y, en consecuencia, hacer una replanificación tan extensa como sea necesario.

Tabla 4.3: Riesgo 02 - Pérdida de datos

R03 - Cambios en requisitos

R03	Cambios en requisitos
Descripción	Durante la evolución del proyecto se ha encontrado un nuevo requisito que debe incorporarse al mismo.
Impacto	Crítico
Probabilidad	Media
Exposición	Bajo
Plan de Protección	Efectuar un análisis de requisitos exhaustivo y realizar reuniones periódicas con los tutores para conseguir el mayor detalle en los requisitos.
Plan de Contingencia	Cumplir con lo estipulado en el plan de control.

Tabla 4.4: Riesgo 03 - Cambios en requisitos

R04 - Fallo de planificación

R04	Fallo de planificación
Descripción	La planificación pensada para el proyecto falla y se demora mucho en el tiempo.
Impacto	Crítico
Probabilidad	Alta
Exposición	Alto
Plan de Protección	Estimar con un margen de error, sin subestimar tecnologías que no se dominan por completo. Tratar de cumplir los plazos de la manera más estricta posible.
Plan de Contingencia	Cumplir con lo estipulado en el plan de control.

Tabla 4.5: Riesgo 04 - Fallo de planificación

R05 - Problemas con el software de terceros

R05	Problemas con el software de terceros
Descripción	La utilización de software realizado por personas ajenas al proyecto pueden presentar problemas no previstos en el desarrollo del mismo.
Impacto	Crítico
Probabilidad	Media
Exposición	Moderado
Plan de Protección	N/A
Plan de Contingencia	Tratar de solucionar el problema y, si no fuera posible, tratar de eludirlo. Como último recurso, considerar la exclusión de la funcionalidad que presente el problema o buscar una alternativa.

Tabla 4.6: Riesgo 05 - Problemas con el software de terceros

R06 - Fallos en implementación

El software desarrollado presenta bugs.

R06	**Fallos en implementación**
Descripción	La utilización de herramientas realizadas por personas ajenas al proyecto presenta problemas no previstos en el desarrollo del mismo.
Impacto	Crítico
Probabilidad	Alta
Exposición	Alto
Plan de Protección	Utilizar buenas prácticas durante la codificación, manteniendo un código fácil de leer y ordenado, menos propenso a errores. Mantener un plan de pruebas conjunto y realizarlo al terminar cada nueva funcionalidad.
Plan de Contingencia	Corregir los bugs y replanificar en caso necesario.

Tabla 4.7: Riesgo 06 - Fallos en implementación

R07 - Exámenes finales

R07	**Exámenes finales**
Descripción	Al estar realizando las tres últimas asignaturas de la carrera a la vez que realizo este proyecto, puede darse el caso de que suspenda alguna asignatura teniendo que asistir a una convocatoria extraordinaria.
Impacto	Medio
Probabilidad	Marginal
Exposición	Bajo
Plan de Protección	Planificar las asignaturas parar llevarlas bien y cuando se acerque el periodo de exámenes dejar de lado el proyecto unos días para poder estudiar.
Plan de Contingencia	En caso de suspenso replanificar el calendario del proyecto.

Tabla 4.8: Riesgo 07 - Exámenes finales

4.4. Presupuesto

Los precios que se aplican a este presupuesto (Figura **4.7**) han sido sacados de las tablas del convenio laboral consultadas en `https://unioninformatica.org/institucional/convenio-colectivo-de-trabajo/`.

Guillermo Muñoz Crespo
Camino Collantes, 5
(34005) Palencia
CIF. 71959391G

Palencia 24 de septiembre de 2020

UD	DESCRIPCIÓN	Precio Oferta	IMPORTE
40,00	Ud. Jornada laboral	159,00 €	6.360,00 €
1,00	Equipo de trabajo (ordenador, pantalla y periféricos)	1.200,00 €.	1.200,00 €.
1,00	Gastos generales 15% (agua, luz, comida)	950,00 €	950,00 €
	IMPORTE TOTAL SIN IVA		8.514,00 €

Fdo. Guillermo Muñoz Crespo

Figura 4.7: Presupuesto final por el desarrollo software

Capítulo 5

Modelo de análisis

En este capítulo vamos a analizar la aplicación, utilizando los requisitos del sistema previamente estudiados.

5.1. Casos de uso del sistema

En este apartado vamos a enfocarnos en el estudio de los casos de uso. Un caso de uso, es un conjunto de escenarios que tienen una meta de usuarios en común [22]. Los casos de uso permiten describir al sistema, el entorno del sistema y su interacción entre el sistema y ese entorno.

Los casos de uso que nos podemos encontrar en este sistema son los que aparecen representados en la Figura **5.1**. Los casos de uso en color azul, ya están desarrollados por el sistema. Los de color rojo no se han podido implementar porque ha quedado fuera de la planificación. Los de color de amarillo, son los casos de uso que se van a realizar.

Como los casos de uso que vamos a definir son los de color amarillo y se realizan tanto por los actores *Profesor* y *Alumno*, cuando hablemos del actor que realiza la acción en estos casos de usos, nos referiremos al actor como actor *Usuario*.

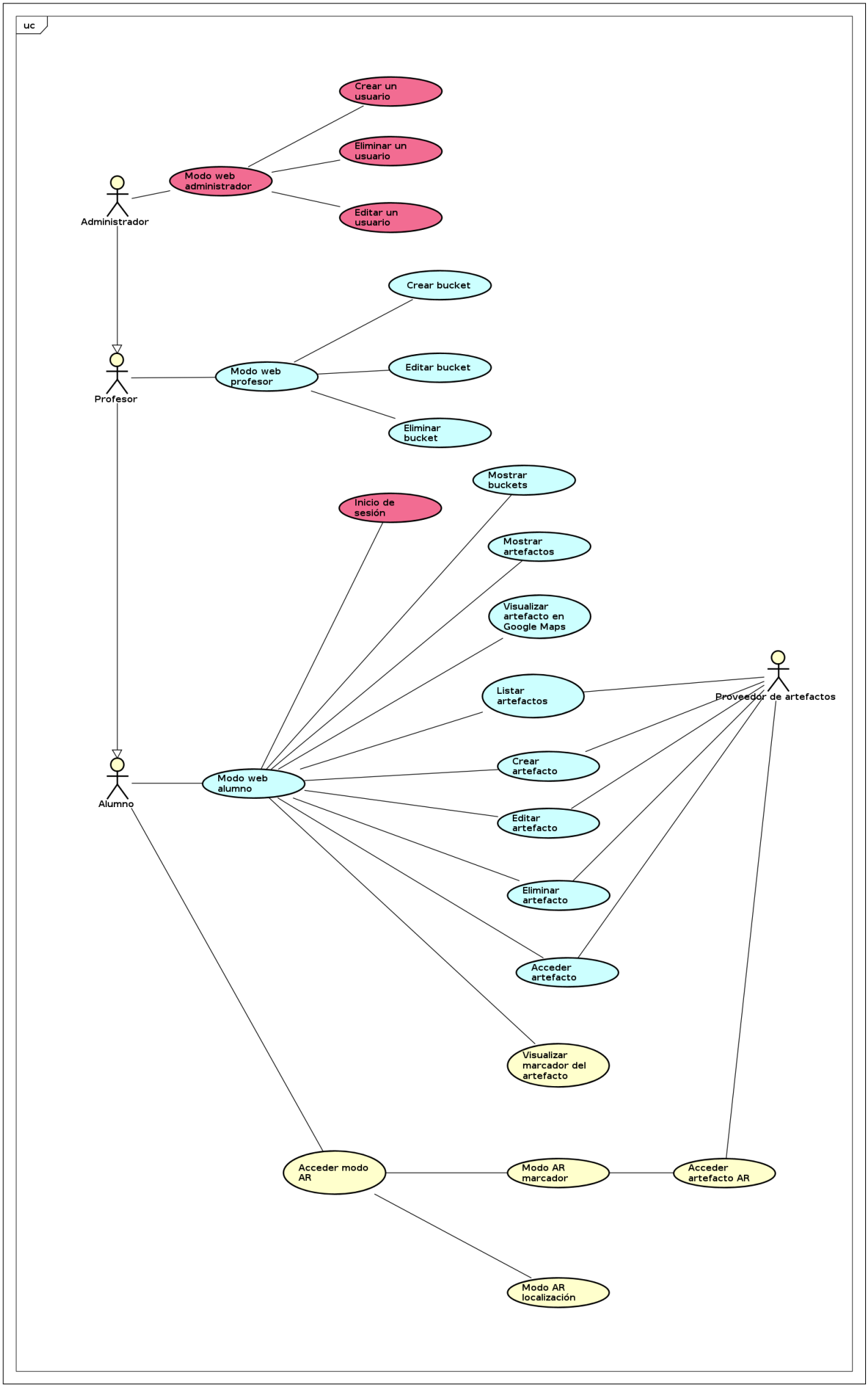


Figura 5.1: Diagrama de casos de uso iniciales

5.1.1. Visualizar marcadores de los artefactos

CU-1.1	**Visualizar marcadores de los artefactos**
Actor	**Alumno, Profesor o Administrador**
Descripción	El sistema deberá permitir al Actor ver todos los marcadores de los artefactos dentro del bucket.
Precondición	El Actor debe estar dentro del modo web Alumno
Secuencia Normal	
Paso	**Acción**
1	El Actor selecciona la opción de visualizar marcadores.
2	El Sistema muestra un listado con todos los marcadores de los artefactos. El caso de uso termina.
Postcondición	
Excepciones	
Variación	**Acción**
2b	Si no hay ningún marcador relacionado con ningún artefacto se mostrará un listado vacío.

Tabla 5.1: Caso de Uso 1.1 - Visualizar marcadores de los artefactos

5.1.2. Modo AR

CU-2.1	**Acceder modo AR**
Actor	**Alumno, Profesor o Administrador**
Descripción	El sistema deberá permitir al Actor acceder al modo de realidad aumentada.
Precondición	Dentro del *bucket* tiene que existir algún artefacto con un marcador o con coordenadas GPS.
Secuencia Normal	
Paso	**Acción**
1	El Actor selecciona el modo AR o escanea un código QR para acceder al modo AR.
2	El Sistema le redirecciona a la vista de realidad aumentada. El caso de uso termina.
Postcondición	
Excepciones	
Variación	**Acción**

Tabla 5.2: Caso de Uso 2.1 - Acceder modo AR

CU-2.2	Modo AR marcador
Actor	**Alumno, Profesor o Administrador**
Descripción	El sistema deberá permitir al Actor seleccionar el modo AR marcador para poder visualizar los artefactos en AR que han sido posicionados a través de un marcador.
Precondición	El Actor debe estar dentro del modo AR.
Secuencia Normal	
Paso	**Acción**
1	El Actor selecciona el modo AR marcador.
2	El Sistema se actualiza para poder reconocer los marcadores.
3	El Actor enfoca con la cámara de su dispositivo al marcador.
4	El Sistema analiza el marcador y comprueba que coincida con un artefacto.
5	El Sistema actualiza la vista del dispositivo mostrando el artefacto en realidad aumentada. El caso de uso termina.
Postcondición	
Excepciones	
Variación	**Acción**
3b	Si el Actor decide retornar al *bucket* accediento a él.
3.1b	El sistema se actualiza y redirecciona al actor a la vista web del *bucket*.
5b	Si el sistema comprueba que el marcador no coincide con el de ningún artefacto, no se mostrará ningún artefacto en realidad aumentada.

Tabla 5.3: Caso de Uso 2.2 - Modo AR marcador

CU-2.3	Acceder artefacto AR
Actor	**Alumno, Profesor o Administrador**
Descripción	El sistema deberá permitir al Actor acceder al artefacto cuando este lo visualice en realidad aumentada.
Precondición	El Actor debe estar dentro del modo AR marcador y estar visualizando el artefacto en realidad aumentada.
Secuencia Normal	
Paso	**Acción**
1	El Actor visualiza el artefacto en realidad aumentada.
2	El Actor accede al artefacto.
3	El Sistema redirige al actor al artefacto.
4	El Actor puede ver el contenido del artefacto. El caso de uso termina.
Postcondición	
Excepciones	
Variación	**Acción**

Tabla 5.4: Caso de Uso 2.3 - Acceder artefacto AR

CU-2.4	Modo AR localización
Actor	**Alumno, Profesor o Administrador**
Descripción	El sistema deberá permitir al Actor seleccionar el modo AR localización para poder visualizar los artefactos en AR que han sido posicionados a través de coordenadas GPS.
Precondición	El Actor debe estar dentro del modo AR.
Secuencia Normal	
Paso	**Acción**
1	El Actor selecciona el modo AR localización.
2	El Sistema se actualiza para poder reconocer las coordenadas GPS.
3	El Actor enfoca con la cámara de su dispositivo en unas coordenadas GPS.
4	El Sistema comprueba que esas coordenadas GPS coincidan con las de un artefacto.
5	El Sistema actualiza la vista del dispositivo mostrando el artefacto en realidad aumentada. El caso de uso termina.
Postcondición	
Excepciones	
Variación	**Acción**
3b	Si el Actor decide retornar al *bucket* accediento a él.
3.1b	El sistema se actualiza y redirecciona al actor a la vista web del *bucket*.
5b	Si el sistema comprueba que las coordenadas GPS no coinciden con las de ningún artefacto, no se mostrará ningún artefacto en realidad aumentada.

Tabla 5.5: Caso de Uso 2.4 - Modo AR localización

5.2. Modelo de dominio

En primer lugar, la aplicación *BucketServer* al tratarse de una aplicación ya creada, tiene su propio modelo de dominio ya definido. Este modelo de dominio se puede observar en la Figura **5.2** [25]. No ha sido necesaria la modificación del modelo de dominio que ya existía, para crear las funcionalidades AR para la aplicación *BucketServer*.

Vamos a realizar una descripción del modelo de dominio de la aplicación *BucketServer*. Para hacer posible el posicionamiento y el acceso a artefactos y *buckets* en diferentes espacios físicos y virtuales, se utiliza la clase *POI*. Sus atributos describen el espacio físico o virtual donde va a ser ubicado. Las clases *Learning bucket* y *Learning artifact* heredan de la clase *POI*, de forma que ambos puedan posicionarse en dichos espacios.

La clase *Learning bucket* puede contener *Learning artifact*. Un *Learning artifact* puede ser de diferentes tipos, estos dependen de la configuración de proveedores de artefactos de la aplicación *BucketServer*, un ejemplo de un proveedor de artefactos para *BucketServer* es *GLUE!* que permite crear un *Google Doc*. El elemento ArtifactType identifica el tipo de artefacto, cuál es su proveedor, y los campos de configuración necesarios para configurar dicho artefacto.

La clase *Learning bucket* dispone de atributos como *visibility*, para que el profesor pueda hacer que el *bucket* sea visible o no, además de otros atributos que definen las posibilidades del *bucket*.

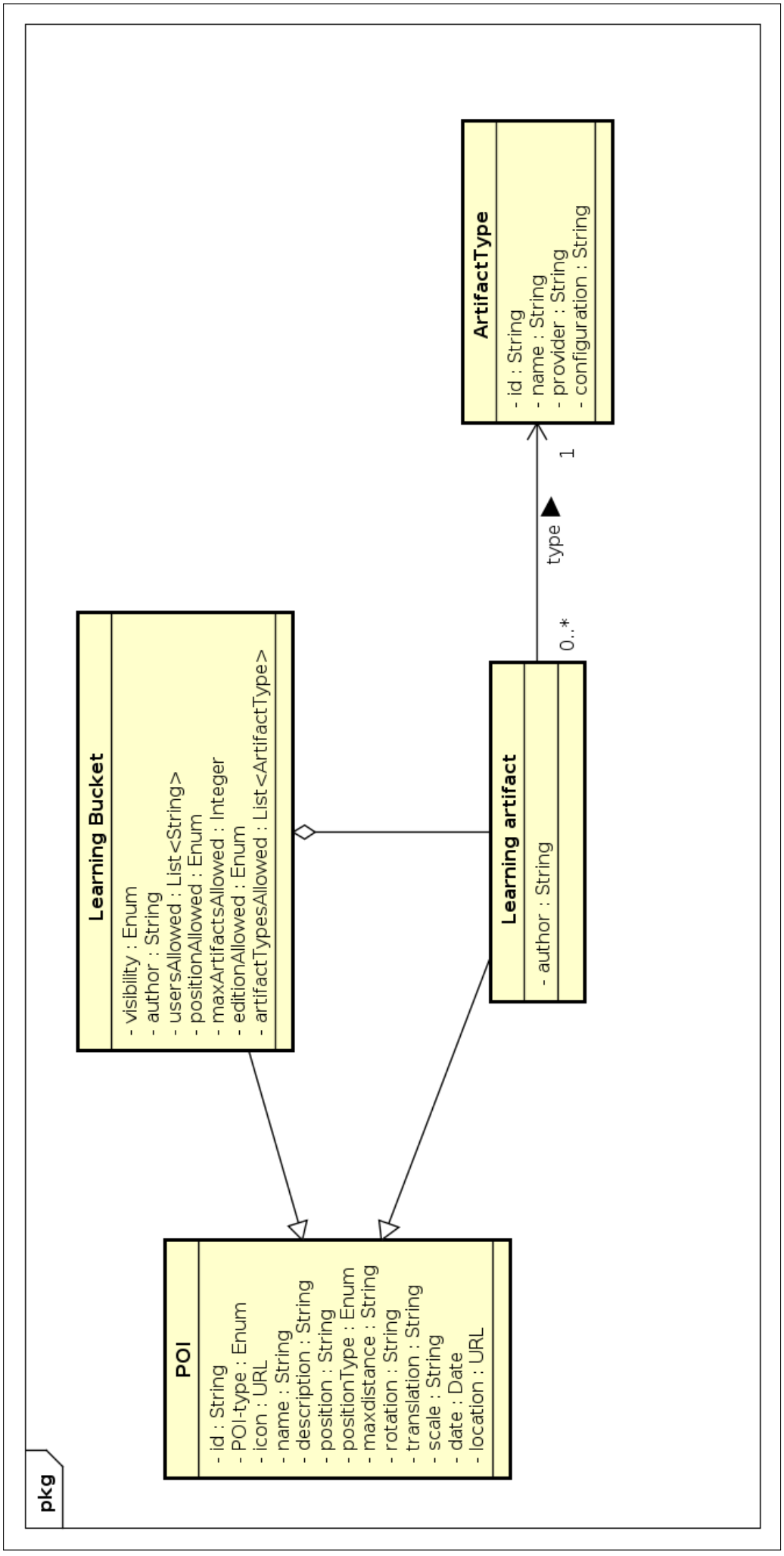


Figura 5.2: Modelo de dominio [25]

Capítulo 6

Diseño

En este capítulo vamos a hablar de las decisiones del diseño, la arquitectura software del sistema, junto con cada componente y cómo están relacionados. Este diseño se realiza teniendo en cuenta las restricciones y requisitos de capítulos anteriores.

6.1. Arquitectura del sistema

Vamos a explicar la arquitectura del sistema que había en *BucketServer* y cuál es la arquitectura propuesta con dos Figuras **6.1** y **6.2**.

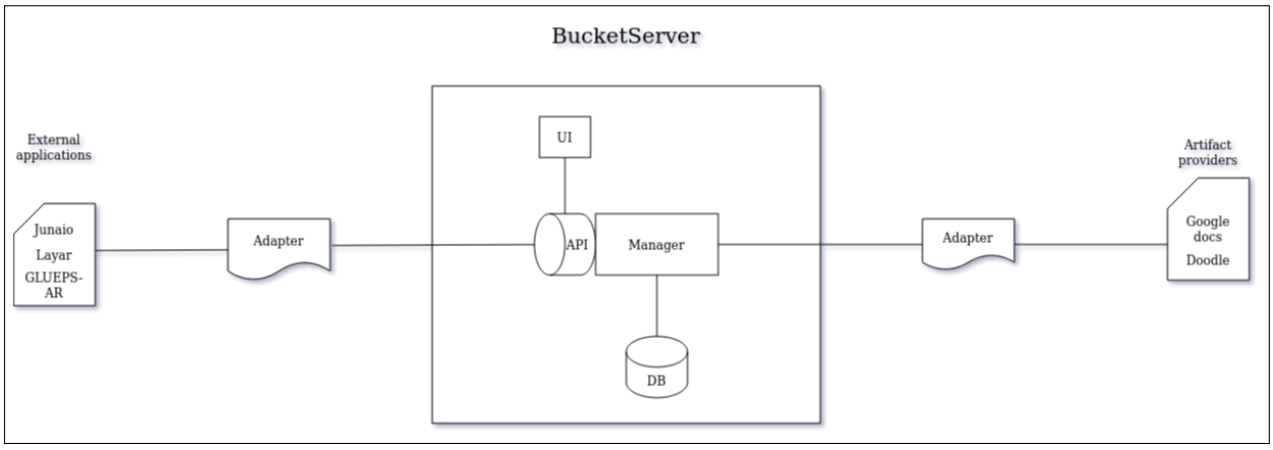


Figura 6.1: Arquitectura del sistema *BucketServer* que existía [25]

El *manager* es el controlador del sistema responsable de administrar los *buckets*, sus artefactos y almacenar la información en la base de datos. El *manager* proporciona una *API* para la comunicación con *external aplications*. También se comunica con *artifacts providers* a través de otra capa de adaptadores. Estos adaptadores se encargan de estandarizar las operaciones de administración sobre los distintos *artifacts providers*, de tal manera que el *manager* siempre pueda utilizar el mismo conjunto de instrucciones definidas, independientemente de la *API* de cada *artifact provider*. *BucketServer* tiene una interfaz de usuario (UI), que actúa como cliente del *manager*, utilizando la API para interactuar con él.

Dentro de la parte de *external aplications* es donde se encontraba la parte de realidad aumentada de *BucketServer*. El problema es que las aplicaciones AR quedaron anticuadas y hubo que buscar una nueva forma de implementar la parte de realidad aumentada. La Figura **6.2** muestra la arquitectura propuesta del sistema *BucketServer* como solución.

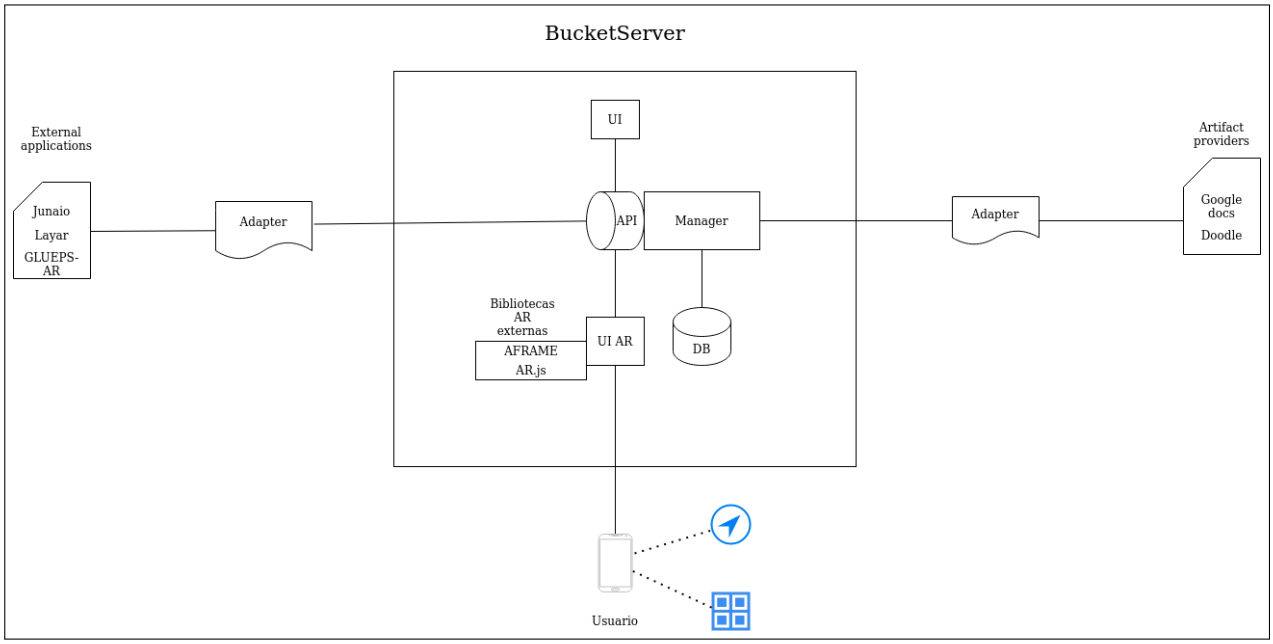


Figura 6.2: Arquitectura del sistema *BucketServer* propuesto

La nueva interfaz de usuario (UI AR) va a contener la parte de realidad aumentada dentro del propio *BucketServer*, utilizando las librerías externas de realidad aumentada de AFRAME y AR.js. Esta UI AR actúa como cliente para los usuarios que accedan, desde dispositivos móviles, para visualizar los artefactos de un *bucket* que estén posicionados por localización o por un marcador, como muestra la Figura **6.2** con los dos iconos azules.

6.1.1. Arquitectura del servidor

La Figura **6.3** se puede observar la arquitectura en capas del servidor. La capa *manager controller* se encarga de comunicarse con las capas *DB*, *model* y *adapter artifacts provider*. En la capa *model* se encuentran las clases definidas del modelo de dominio.

La capa *API* se encargar de recibir y redirigir las peticiones que llegan al servidor desde las capas *UI*, *UI AR* y *adapter external aplications*, hacia otras capas para poder resolver estas peticiones.

Las capas *adapter external aplications* y *adapter artifacts provider* son las capas cuya función es comunicarse con las aplicaciones externas al servidor. Las capas *UI* y *UI AR* son las interfaces de usuario, cuya función es mostrar un entorno gráfico para que el usuario pueda utilizar la aplicación.

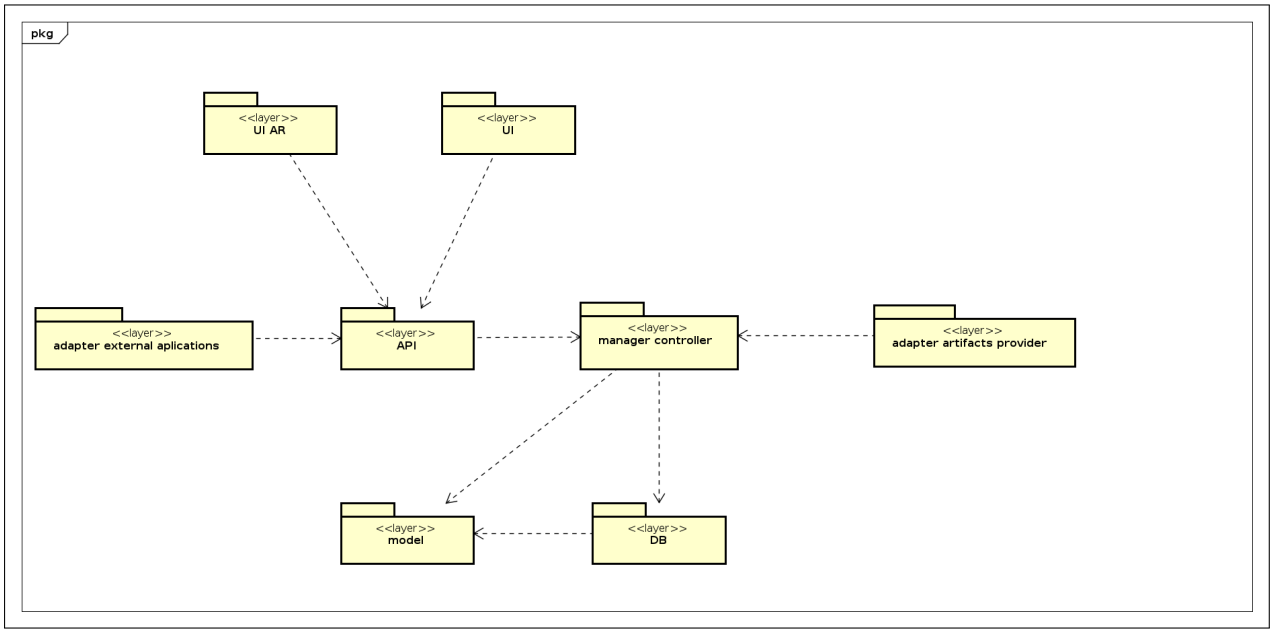


Figura 6.3: Arquitectura servidor

6.1.2. Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar la disposición física de los artefactos software en nodos. El objetivo de estos diagramas es mostrar las relaciones físicas entre los componentes software y hardware en el sistema.

En la Figura **6.4**, se muestra el diagrama de despliegue del sistema. En él se puede observar cómo hay un cliente que se conecta a la página web, por medio de una petición *HTTPS* hacia el servidor web *Apache*. Este, al recibir la petición, utiliza una conexión *proxy reverse* con el servidor *Java BucketServer*, cuya petición se encargará el nuevo servidor.
El server *Java BucketServer* se encargará del acceso a la base de datos o de comunicarse con los servicios externos de los servidores de *external aplications* y *artifact providers*, utilizando los *adapters*. Una vez realiza todas las operaciones, el *server BucketServer* devuelve la petición al servidor web *Apache*, para que le de una respuesta al cliente en la página web.

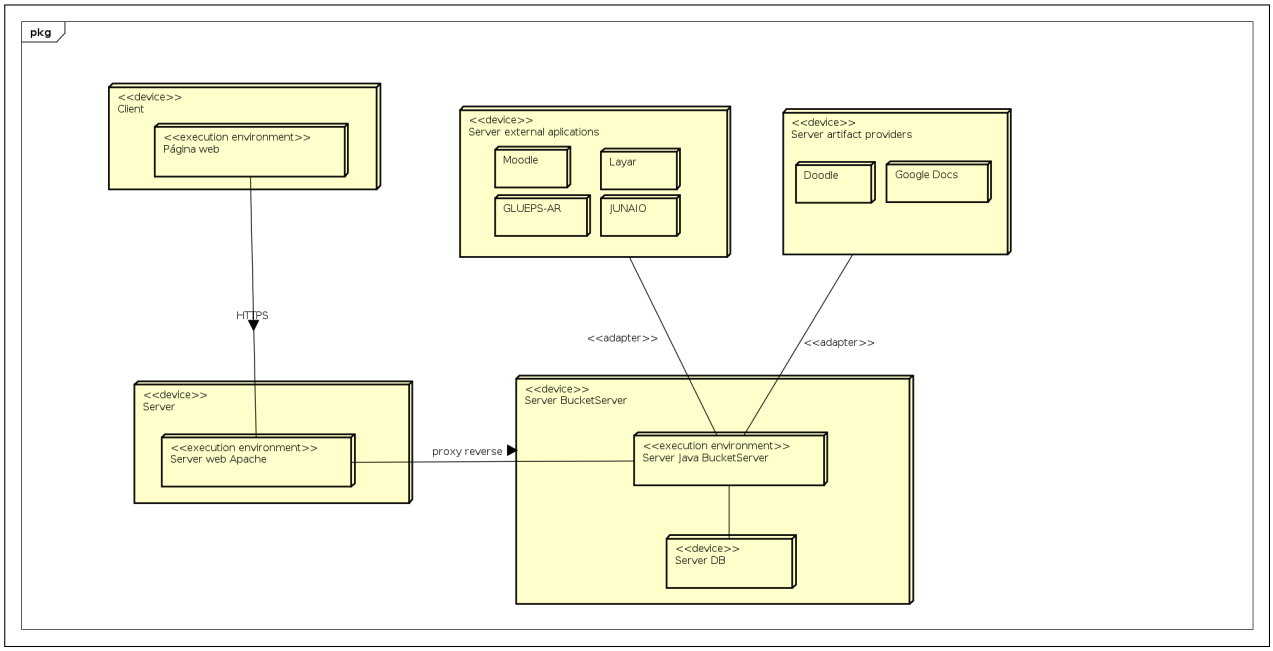


Figura 6.4: Diagrama de despliegue del sistema

6.1.3. Arquitectura cliente

Como la aplicación *BucketServer* ya estaba desarrollada, nos hemos enfocado en la integración de las nuevas funcionalidades en el *frontend*. Del *backend* casi no se han realizado cambios.
El *frontend* se ha desarrollado con el *framework Dojo Toolkit* que se había utilizado para el desarrollo del *BucketServer*, utilizando *HTML*, *CSS*, *JavaScript*, además de librerías externas de *JavaScript*.

6.2. Patrón Arquitectónico

El patrón arquitectónico desarrollado para el sistema *BucketServer* es el patrón *MVC* (*Modelo Vista y Controlador*).

Para la nueva parte que hay que diseñar se ha tenido en cuenta el mismo patrón arquitectónico para su desarrollo. Como el trabajo a realizar es en el *frontend*, solo utilizaremos la *Vista* y el *Controlador* del patrón *MVC*. Los controladores gestionan la entrada del usuario y se comunicará con el *Modelo* y la *Vista*. La *Vista* muestra la información que se envía al cliente.

Existen dos nuevos controladores:

- Controlador modo AR marcador: Este controlador se encargará del acceso a los artefactos, que tengan posicionamiento de marcadores. Reconocimiento de marcadores, acceso al artefacto, creación, activación y desactivación.
- Controlador modo AR localización: Este controlador se encargará del acceso a los artefactos, que tengan posicionamiento de geoposición. Reconocimiento de coordenadas, activación y desactivación.

Existen dos nuevas vistas:

- Vista modo AR marcador: Esta vista se encargará de mostrar la información en realidad aumentada de los artefactos, con posicionamiento de tipo marcador.
- Vista modo AR localización: Esta vista se encargará de mostrar la información en realidad aumentada de los artefactos, con posicionamiento de geoposición.

6.3. Diseño de la interfaz de usuario

En esta sección se incluyen los bocetos de las nuevas pantallas del modo AR localización y modo AR marcador.

6.3.1. Interfaz modo AR marcador

Esta interfaz es de un usuario con su dispositivo móvil, muestra por pantalla la cámara del móvil y enfoca un marcador para mostrar un artefacto en realidad aumentada. Ver Figura **6.5**.

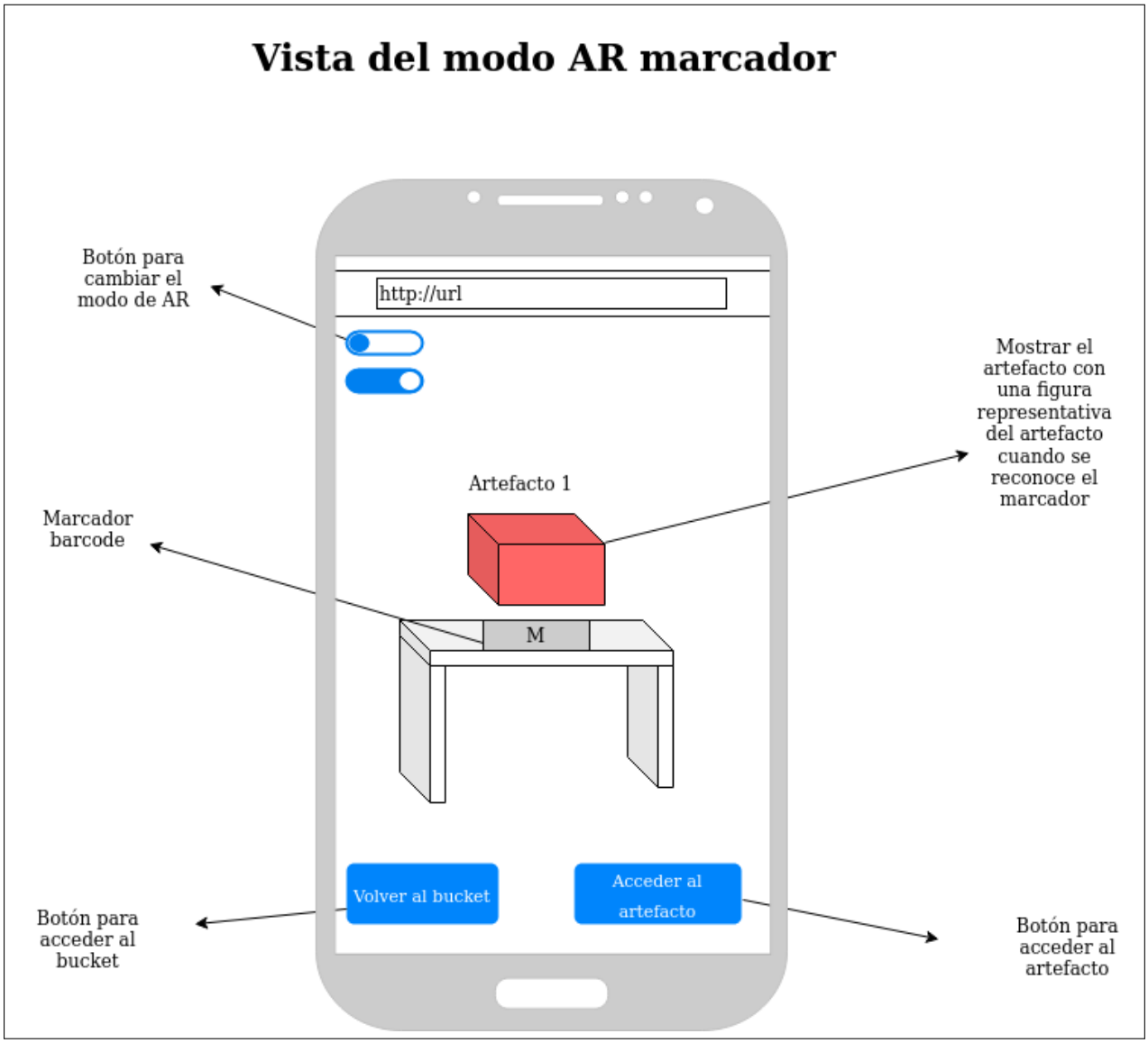


Figura 6.5: Interfaz modo AR marcador

6.3.2. Interfaz modo AR localización

Esta interfaz es de un usuario que esta en un espacio público y accede con su dispositivo móvil, mostrando por pantalla la cámara del móvil los artefactos que se encuentran cerca en realidad aumentada. Ver Figura **6.6**.

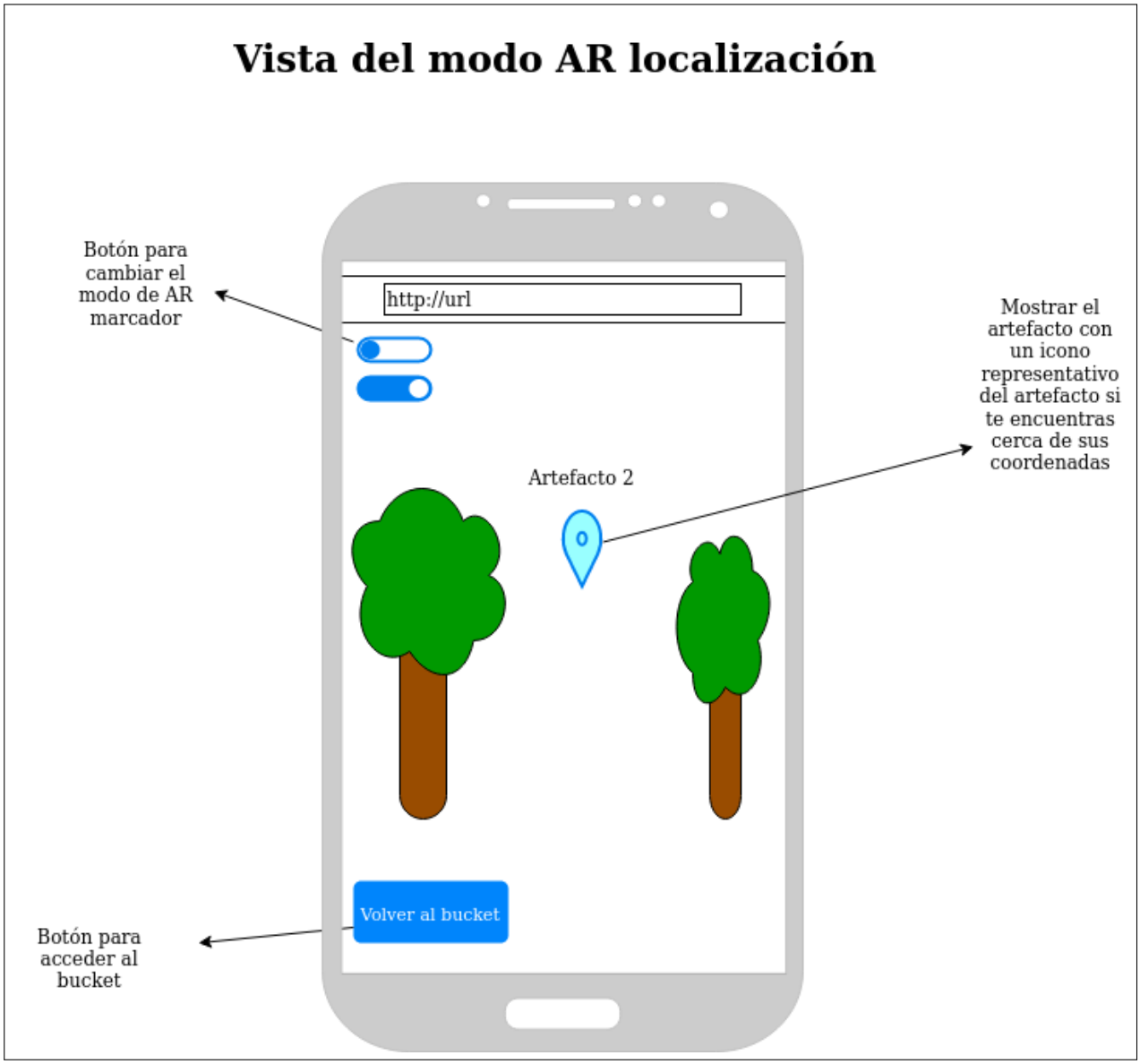


Figura 6.6: Interfaz modo AR localización

6.3.3. Interfaz modo web

Esta interfaz es de un usuario que esta dentro de un *bucket* en el modo web, mostrando en la pantalla las distintas funciones que puede realizar en este modo. Como se observa en la Figura **6.7**

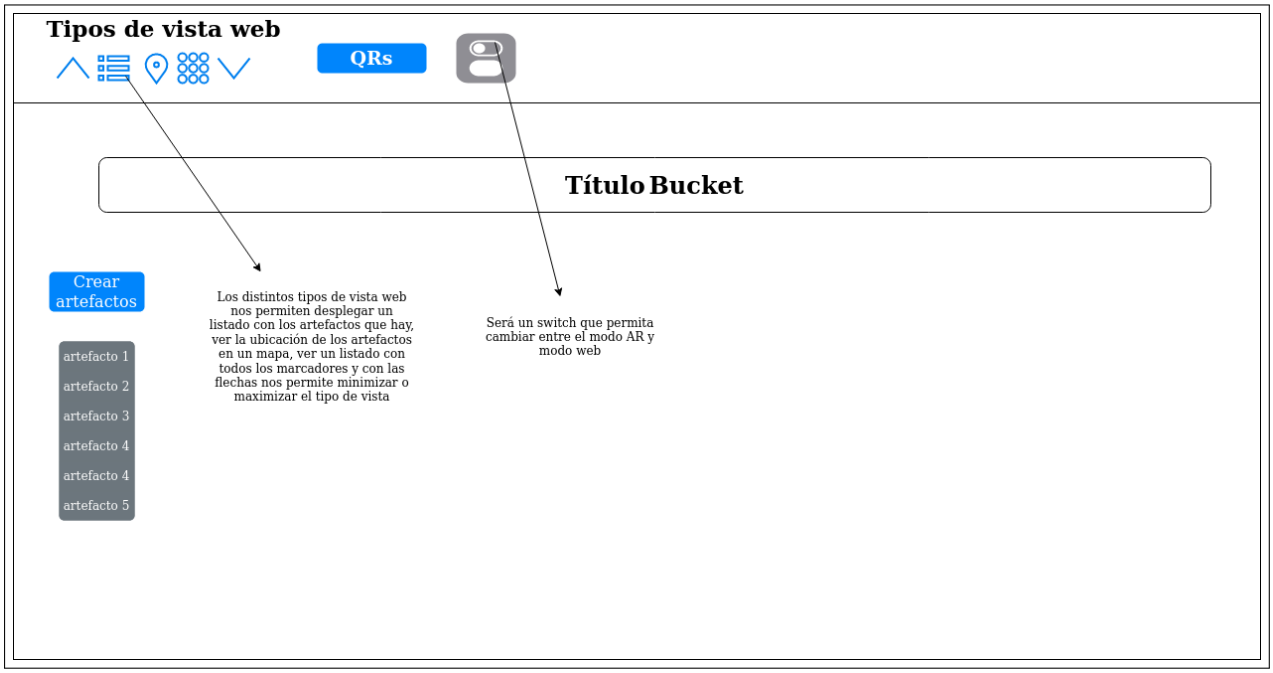


Figura 6.7: Interfaz modo web

6.4. Diagramas de secuencia

6.4.1. Diagrama visualizar marcadores y códigos QR del artefacto

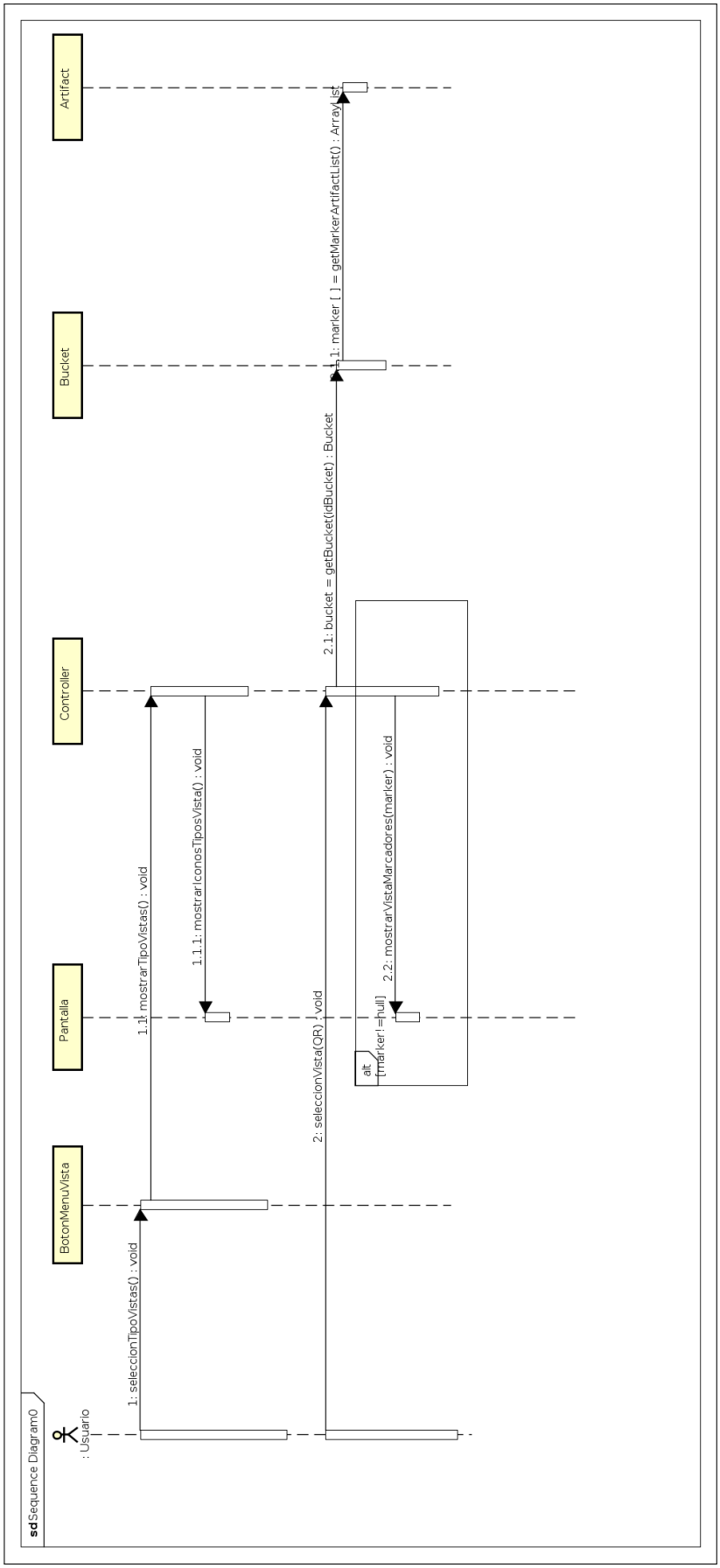


Figura 6.8: Diagrama de secuencia CU: Visualizar marcadores y códigos QR del artefacto

6.4.2. Diagrama leer QR para acceder al modo AR

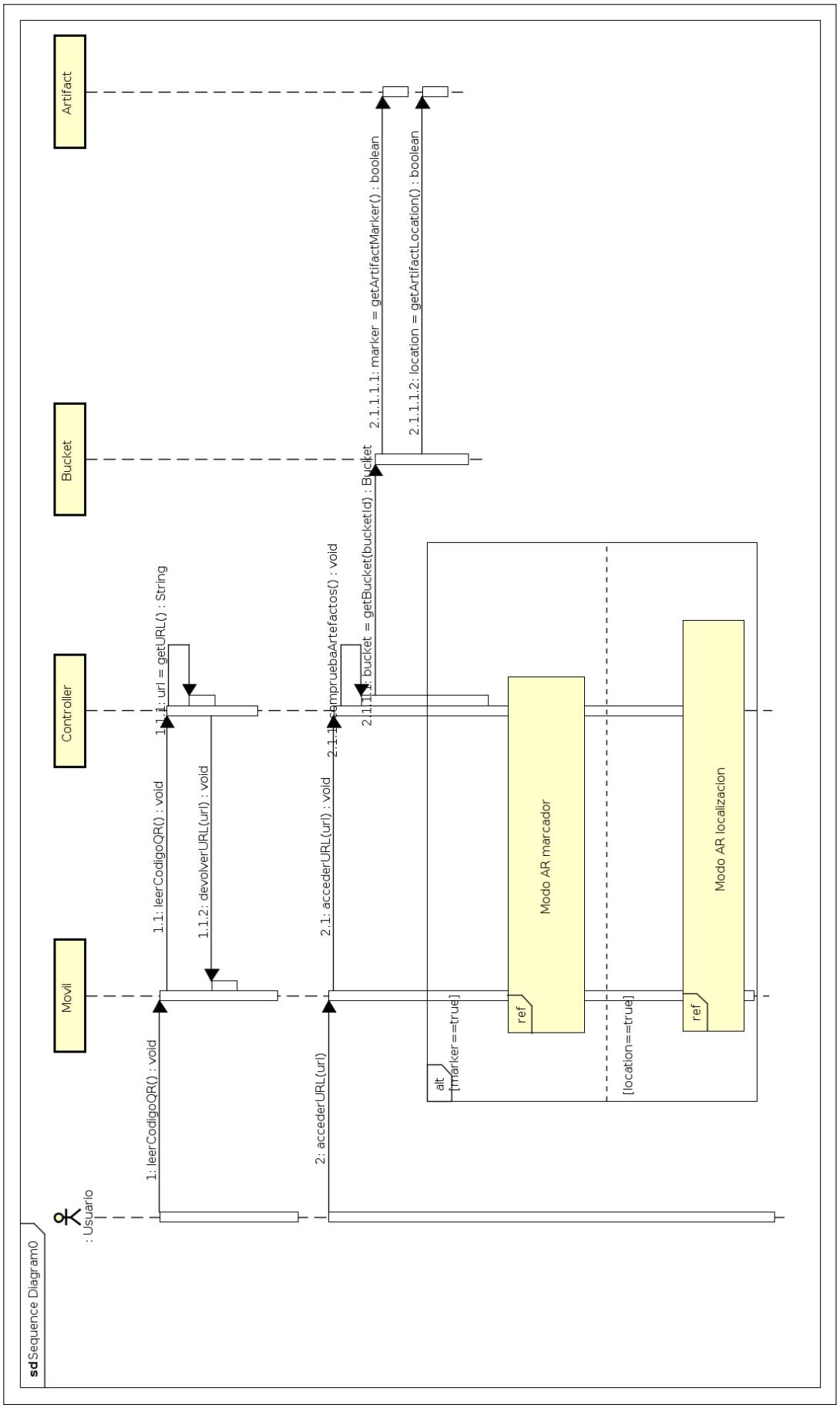


Figura 6.9: Diagrama de secuencia CU: Modo AR

6.4.3. Diagrama modo AR marcador

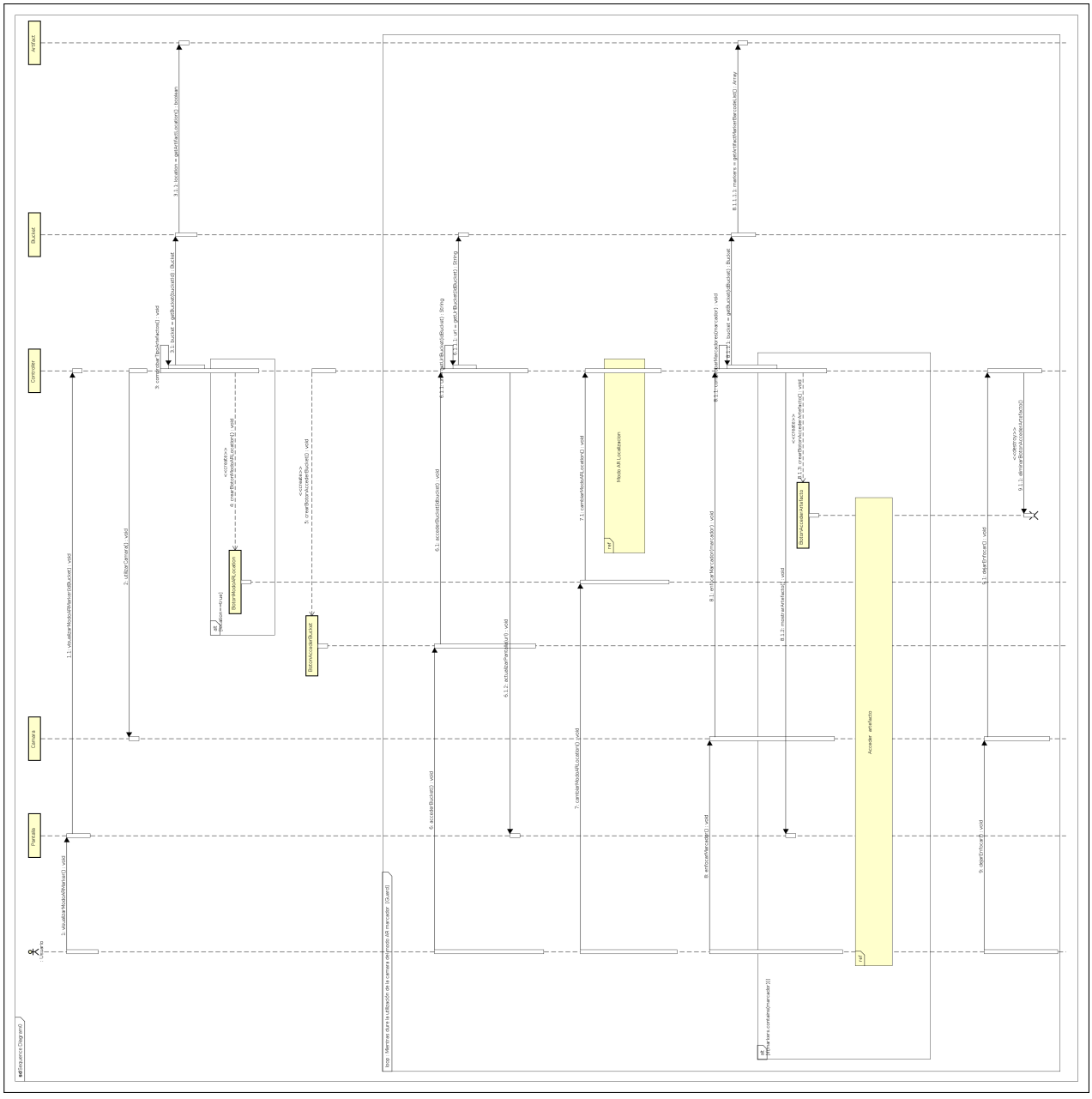


Figura 6.10: Diagrama de secuencia CU: Modo AR marcador

6.4.4. Diagrama modo AR localización

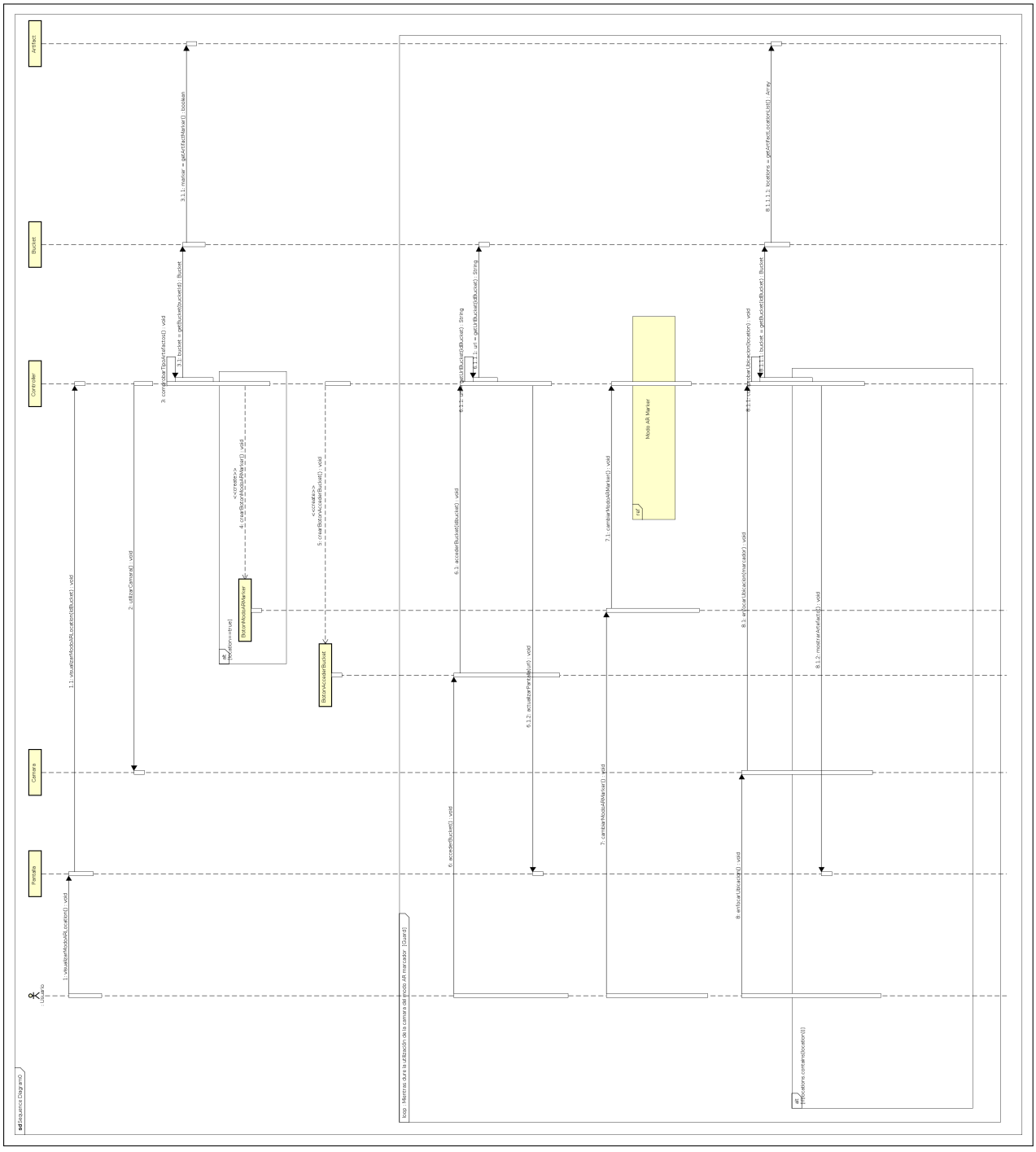


Figura 6.11: Diagrama de secuencia CU: Modo AR localización

6.4.5. Diagrama acceder artefacto

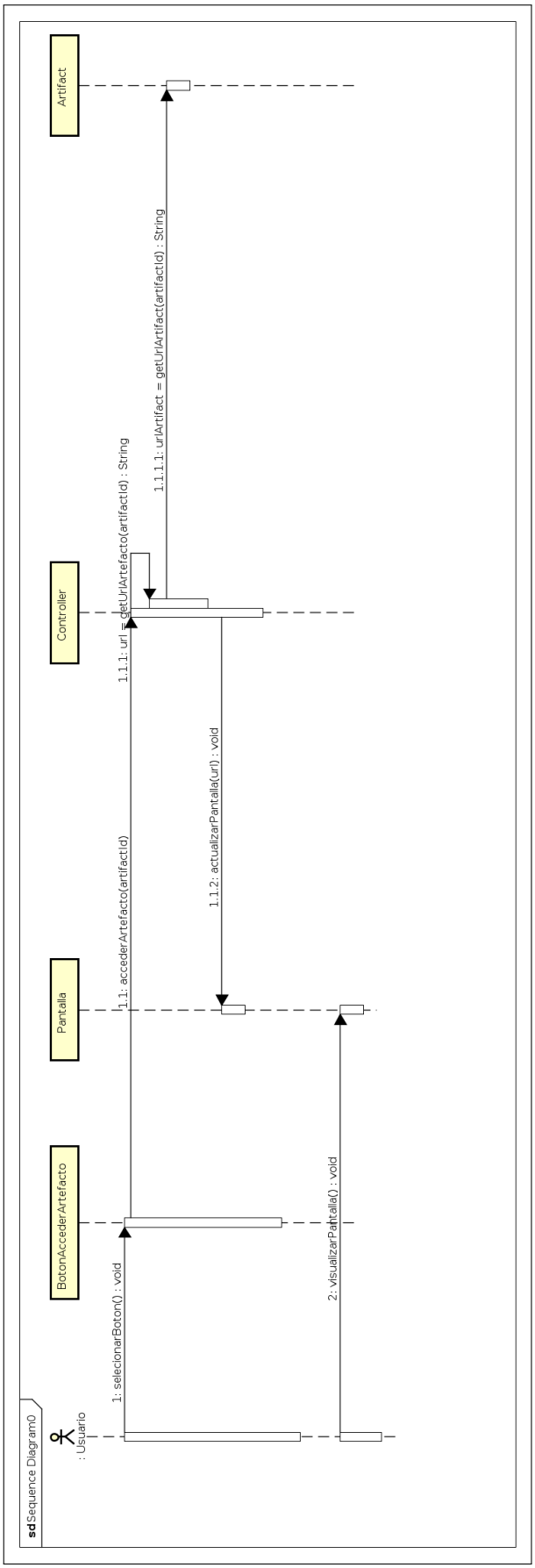


Figura 6.12: Diagrama de secuencia CU: Acceder artefacto AR

Capítulo 7

Implementación

Se detallarán cuestiones prácticas que se consideren importantes sobre la implementación, como el entorno de desarrollo, librerías utilizadas, control de versiones, etcétera.

7.1. Entorno de desarrollo

Para el desarrollo se ha trabajado con un portátil MSI Modern 14 A10M, con un procesador i5 de décima generación, una memoria de 16GB de RAM y un disco duro de 512GB SSD. El sistema operativo instalado en este portátil es Ubuntu 18.0.4.

Para el desarrollo *software* se ha trabajado con el IDE Visual Studio Code (VSCode) [1]. Este IDE posee una gran variedad de extensiones que se pueden instalar para agilizar el desarrollo software, algunas de estas extensiones que hemos usado son para la programación en Java, HTML, CSS y Javascript.

7.2. Herramientas utilizadas

Las herramientas utilizadas durante el desarrollo han sido las siguientes:

- Astah [2]: Herramienta utilizada para la creación de diagramas *software*
- SmartGit [3]: Herramienta gráfica que permite la conexión con aplicaciones de control de repositorios, en este caso *Github*.
- MySQL Workbench [4]: Permite la conexión con el sistema gestor de base de datos MySQL para realizar operaciones en la misma.
- Draw.io[5]: Herramienta utilizada para la creación de esquemas y diagramas.

7.3. Control de versiones

El sistema utilizado para la gestión de proyectos y control de versiones de código ha sido *Github*. La metodología de trabajo seguida con este sistema se muestra en la Figura **7.1**.

[1]Entorno de desarrollo: `https://code.visualstudio.com/`
[2]Editor para diseño *software*: `https://astah.net/`
[3]Entorno gráfico de control de repositorios: `https://www.syntevo.com/smartgit/`
[4]Entorno gráfico para MySQL: `https://www.mysql.com/products/workbench/`
[5]Editor de diagramas y esquemas: https://app.diagrams.net/

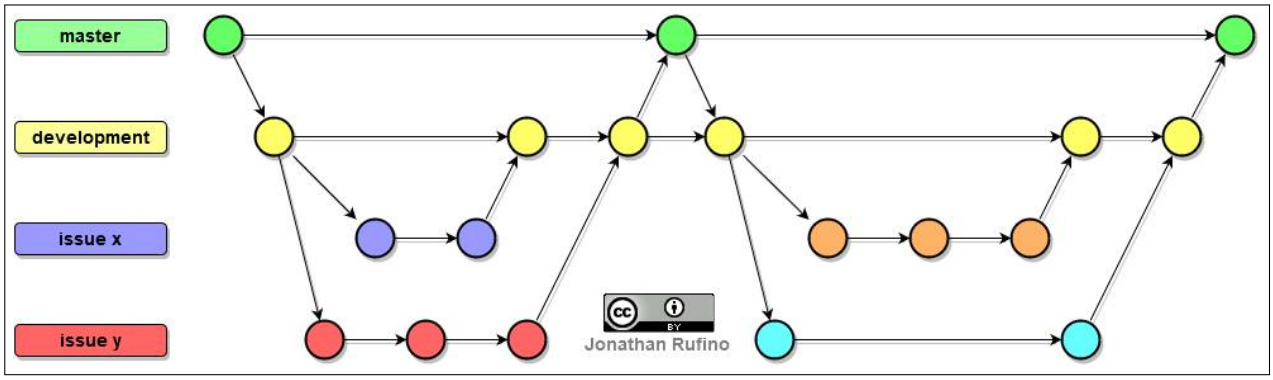


Figura 7.1: Utilización de ramas Git [19]

Como se puede observar en la figura anterior, para cada nueva funcionalidad de la aplicación que se iba a desarrollar, se creaba una rama con el nombre de la funcionalidad. Una vez finalizaba el desarrollo de esa funcionalidad, se incorporaba el contenido de esa rama a la rama de *development*. Una vez se termina todo el desarrollo *software* se fusiona la rama *development* a la rama *master*.

7.4. Librerías utilizadas

Las librerías externas utilizadas de *Javascript* para la realidad aumentada, han sido las siguientes:

- **AR.js**
  - `https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js`
  - `https://aframe.io/releases/1.0.4/aframe.min.js`
  - `https://unpkg.com/aframe-look-at-component@0.8.0/dist/aframe-look-at-component.min.js`
  - `https://raw.githack.com/AR-js-org/AR.js/master/three.js/build/ar-nft.js`

Estas librerías permiten que cuando un marcador es encontrado, pueda mostrar algún contenido encima del marcador. Además de mostrar contenido de realidad aumentada en el dispositivo del usuario, dependiendo de su geoposición.

7.5. Servidor utilizado

Para la implementación es necesario la utilización de un servidor *Apache HTTP Server*[6]. Es un software de servidor web gratuito y de código abierto para plataformas Unix. Mantenido y desarrollado por *Apache Software Foundation*
Se necesita la utilización de *Apache HTTP Server* para la aplicación *BucketServer*. El servidor *Apache HTTP Server* sirve para mostrar el contenido web a través del navegador, proporcionar una conexión segura SSL, para que funcione correctamente el acceso a la cámara del dispositivo que necesitan las librerías *Javascript* y para realizar una conexión *proxy reverse*, con el servidor *Java* del *backend* de la aplicación *BucketServer*.

[6]Apache HTTP Server - Servidor web: `https://httpd.apache.org/`

7.6. Base de datos utilizada

La aplicación *BucketServer* utiliza como sistema gestor de base de datos MySQL[7]. Es un gestor de bases de datos relacional y de código abierto desarrollado por Oracle. La versión instalada es MySQL Server 5.7.

[7]MySQL - Gestor de base de datos: `https://www.mysql.com/`

Capítulo 8

Plan de pruebas y validación

En este capítulo se detallan las pruebas realizadas para comprobar el correcto funcionamiento de las funcionalidades implementadas.

8.1. Pruebas vista marcadores y QRs

Prueba 1.1	
Descripción	Esta prueba está destinada a probar el funcionamiento de la vista, que muestra una lista de los marcadores *barcode* y códigos QR de los artefactos
Acción	Acceder a la vista de marcadores y códigos QR sin existir ningún artefacto.
Resultado Esperado	No se muestra ningún marcador o QR en la lista.
Resultado Obtenido	No se muestra ningún marcador o QR en la lista.

Prueba 1.2	
Descripción	Esta prueba está destinada a probar el funcionamiento de la vista, que muestra una lista de los marcadores *barcode* y códigos QR de los artefactos
Acción	Crear un artefacto posicionándolo en un código QR, acceder a la vista de marcadores y códigos QR.
Resultado Esperado	Se muestra el código QR del artefacto.
Resultado Obtenido	Se muestra el código QR del artefacto.

Prueba 1.3	
Descripción	Esta prueba está destinada a probar el funcionamiento de la vista, que muestra una lista de los marcadores *barcode* y códigos QR de los artefactos
Acción	Crear un artefacto posicionándolo en un marcador, acceder a la vista de marcadores y QRs.
Resultado Esperado	Se muestra el marcador *barcode* del artefacto.
Resultado Obtenido	Se muestra el marcador *barcode* del artefacto.

8.2. Creación código QR para acceder al modo AR

Prueba 2.1	
Descripción	Esta prueba está destinada a probar el funcionamiento del código QR para acceder al modo AR.
Acción	Accedemos a un *bucket* en el que no hay ningún artefacto con el posicionamiento de marcador o geoposición.
Resultado Esperado	No se muestra ningún código QR para acceder al modo AR.
Resultado Obtenido	No se muestra ningún código QR para acceder al modo AR.

Prueba 2.2	
Descripción	Esta prueba está destinada a probar el funcionamiento del código QR para acceder al modo AR.
Acción	Accedemos a un *bucket* y creamos un artefacto con el posicionamiento de marcador. Escaneamos el código QR con el móvil y accedemos a la URL.
Resultado Esperado	Accedemos al modo AR.
Resultado Obtenido	Accedemos al modo AR.

Prueba 2.3	
Descripción	Esta prueba está destinada a probar el funcionamiento del código QR para acceder al modo AR.
Acción	Accedemos a un *bucket* y creamos un artefacto con el posicionamiento de geoposición. Escaneamos el código QR con el móvil y accedemos a la URL.
Resultado Esperado	Accedemos al modo AR.
Resultado Obtenido	Accedemos al modo AR.

8.3. Pruebas modo AR marcador

Prueba 3.1	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Clickamos el botón «volver al bucket».
Resultado Esperado	El sistema nos redirige a la vista del *bucket*.
Resultado Obtenido	El sistema nos redirige a la vista del *bucket*.

Prueba 3.2	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Clickamos el botón «GPS».
Resultado Esperado	El sistema nos redirige al modo AR localización.
Resultado Obtenido	El sistema nos redirige al modo AR localización.

Prueba 3.3	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Enfocamos un marcador que pertenece a un artefacto.
Resultado Esperado	El sistema muestra un cubo que representa el artefacto y un texto con su nombre. Además, muestra un botón para acceder al artefacto.
Resultado Obtenido	El sistema muestra un cubo que representa el artefacto y un texto con su nombre. Además, muestra un botón para acceder al artefacto.

Prueba 3.4	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Enfocamos un marcador que no pertenece a un artefacto.
Resultado Esperado	El sistema no muestra nada.
Resultado Obtenido	El sistema no muestra nada.

Prueba 3.5	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Clickamos el botón «acceder artefacto».
Resultado Esperado	El sistema redirige al usuario a la ubicación del artefacto.
Resultado Obtenido	El sistema redirige al usuario a la ubicación del artefacto.

8.4. Pruebas modo AR localización

Prueba 4.1	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR localización.
Acción	Clickamos el botón «volver al bucket».
Resultado Esperado	El sistema nos redirige a la vista del *bucket*.
Resultado Obtenido	El sistema nos redirige a la vista del *bucket*.

Prueba 4.2	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Clickamos el botón «Marker».
Resultado Esperado	El sistema nos redirige al modo AR marcador.
Resultado Obtenido	El sistema nos redirige al modo AR marcador.

Prueba 4.3	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Enfocamos la cámara del móvil cerca de la ubicación de un artefacto.
Resultado Esperado	El sistema muestra un icono que representa el artefacto y un texto con su nombre.
Resultado Obtenido	El sistema muestra un icono que representa el artefacto y un texto con su nombre.

Prueba 4.4	
Descripción	Esta prueba está destinada a probar el funcionamiento del modo AR marcador.
Acción	Enfocamos la cámara del móvil lejos de la ubicación de un artefacto.
Resultado Esperado	El sistema no muestra nada.
Resultado Obtenido	El sistema no muestra nada.

8.5. Pruebas de interfaz de usuario

Prueba 5.1	
Descripción	Esta prueba está destinada a probar que la aplicación es manejable con una sola mano desde un dispositivo móvil.
Acción	Se realizan diversas operaciones como crear un artefacto, eliminarlo, visualizarlo, con distintos perfiles de usuarios.
Resultado Esperado	Todos los usuarios han sido capaz de realizar las operaciones con una mano.
Resultado Obtenido	Todos los usuarios han sido capaz de realizar las operaciones con una mano.

Prueba 5.2	
Descripción	Esta prueba está destinada a comprobar el porcentaje de errores al utilizar la aplicación.
Acción	Realizamos diez operaciones para crear distintos artefactos, editamos los diez artefactos y después eliminamos los diez artefactos.
Resultado Esperado	No se obtiene ningún error al realizar estas operaciones.
Resultado Obtenido	No se obtiene ningún error al realizar estas operaciones.

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

En la realización de este TFG se han implementado diversas funcionalidades a una aplicación ya existente, *BucketServer*. Se ha sustituido la funcionalidad de AR que utilizaba aplicaciones externas, que habían quedado anticuadas, por una nueva tecnología de realidad aumentada dentro de la aplicación. Los objetivos iniciales han sido cumplidos y para cada uno de ellos se han completado varios hitos:

1. Análisis y selección de tecnologías AR para implementar en la aplicación *BucketServer*:
   - Búsqueda de aplicaciones móviles AR compatibles con la aplicación *BucketServer*.
   - Búsqueda de librerías AR web compatibles con la aplicación *BucketServer*.
2. Implementación de funcionalidades de realidad aumentada, sustituyendo a las aplicaciones externas de *BucketServer*.
   - Implementación *software* para crear artefactos dentro de un *bucket* que se puedan posicionar en marcadores AR o ubicaciones AR, utilizando las librerías de *AR.js*.
   - Código QR para acceder al modo AR, desde cualquier dispositivo capaz de leer códigos QR.
   - Implementación de una vista para la visualización de los marcadores de los artefactos.
   - Modificación del modo web para mejorar la interfaz del usuario con el propósito de conseguir una mayor facilidad a la hora de seleccionar entre los dos modos de visualización y los distintos tipos de vista en el modo web.
3. Actualización de la documentación *BucketServer*
   - Creación de un manual de instalación
   - Creación de un manual de usuario

9.2. Trabajo futuro

La aplicación *BucketServer* puede aumentar sus funcionalidades y mejorar las existentes. Algunas de estas mejoras son:

- Actualización de las funciones de la librerías de *AR.js*, permitiendo que cuando se detecte la ubicación de un artefacto en realidad aumentada, se cree un evento para poder acceder a ese artefacto.

- Implementación de medidas de seguridad, creando roles de usuario para los actores profesores y usuarios.
- Actualización de la aplicación *BucketServer*, utilizando tecnologías actuales para crear un código mas estructurado y legible.

9.3. Valoración personal

Los conocimientos adquiridos en el desarrollo de ese proyecto sobre realidad aumentada, me han hecho reflexionar sobre las posibilidades de esta tecnología en el mundo actual, pudiendo ser una herramienta de gran utilidad para diversos sectores, no solo en el ámbito educativo, además de ser de gran utilidad en otros ámbitos.
La implementación de nuevas funcionalidades dentro de un sistema que ya ha sido creado ha supuesto una gran dificultad técnica. Sin embargo, me ha servido para mejorar mis habilidades y conocimientos como ingeniero informático.

Referencias

[1] API de ARCore. https://developers.google.com/ar/reference/. [Online; accessed 28-March-2020].

[2] AR.js Documentation. https://ar-js-org.github.io/AR.js-Docs/. [Online; accessed 8-September-2020].

[3] Augmented Reality point of interest service environment. https://github.com/ARPOISE/ARpoise/blob/master/README.md. [Online; accessed 26-March-2020].

[4] Documentación de ARToolkit. http://www.artoolkitx.org/. [Online; accessed 27-March-2020].

[5] Documentación EasyAR. https://www.easyar.com/. [Online; accessed 28-March-2020].

[6] Documentación Vuforia. https://developer.vuforia.com/. [Online; accessed 26-March-2020].

[7] Documentación Wikitude. https://www.wikitude.com/external/doc/documentation/studio-api/. [Online; accessed 25-March-2020].

[8] Geoar. https://geoar.it/. [Online; accessed 26-March-2020].

[9] Github de ARToolKit. https://github.com/artoolkitx/artoolkitx/wiki. [Online; accessed 27-March-2020].

[10] Goodbye Blippar, welcome back Layar? https://sifted.eu/articles/goodbye-blippar-welcome-back-layar/. [Online; accessed 25-March-2020].

[11] Guía de Proxy inverso. https://httpd.apache.org/docs/trunk/es/howto/reverse_proxy.html. [Online; accessed 24-July-2020].

[12] How to create POI in ARPoise? https://github.com/ARPOISE/ARpoise/blob/master/README.md. [Online; accessed 26-March-2020].

[13] Kudan. https://www.kudan.io/. [Online; accessed 26-March-2020].

[14] KudanAR. https://www.xlsoft.com/en/products/kudan/index.html. [Online; accessed 26-March-2020].

[15] MaxST. http://maxst.com/#/. [Online; accessed 28-March-2020].

[16] Módulo Apache mod_ssl. https://httpd.apache.org/docs/trunk/es/mod/mod_ssl.html. [Online; accessed 24-July-2020].

[17] Top 10 AR Tools for App Development. https://arvrjourney.com/top-10-ar-tools-for-app-development-6dab56a833db. [Online; accessed 15-March-2020].

[18] Top 5 Free AR Development Tools 2019 For Creating Augmented Reality App. https://yourstory.com/mystory/top-5-ar-development-tools-2019-creating-augmented. [Online; accessed 15-March-2020].

[19] Utiliza las ramas. http://agrega.juntadeandalucia.es/repositorio/20022017/c8/es-an_2017022012_9123136/42_utiliza_las_ramas.html. [Online; accessed 12-September-2020].

[20] Donna R Berryman. Augmented reality: A review. *Medical reference services quarterly*, 31(2):212–218, 2012.

[21] Elliot J. Chikofsky and James H Cross. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1):13–17, 1990.

[22] Martin Fowler. *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[23] Elena Mediavilla. UML: Modelado de casos de uso. https://www.ctr.unican.es/asignaturas/MC_OO/Doc/Casos_de_uso.pdf. [Online; accessed 15-July-2020].

[24] Juan A Muñoz-Cristóbal, Juan I Asensio-Pérez, Alejandra Martínez-Monés, Luis P Prieto, Iván M Jorrín-Abellán, and Yannis Dimitriadis. Bucket-server: A system for including teacher-controlled flexibility in the management of learning artifacts in across-spaces learning situations. In *Design for Teaching and Learning in a Networked World*, pages 518–521. Springer, 2015.

[25] Juan A Muñoz-Cristóbal, Juan I Asensio-Pérez, Alejandra Martínez-Monés, Luis P Prieto, Ivan M Jorrin-Abellan, and Yannis Dimitriadis. Learning buckets: Helping teachers introduce flexibility in the management of learning artifacts across spaces. *IEEE Transactions on Learning Technologies*, 11(2):203–215, 2017.

[26] Ian Sommerville. *Software Engineering*. Pearson Educación, 2005.

Anexos

Anexo I

Manual de instalación

Para poder instalar la aplicación *BucketServer* es necesario seguir los siguientes pasos.

I.1. Requisitos

- Servidor web: Apache HTTP Server versión 2.4.
- Gestor de base de datos: MySQL Server 5.7.
- Instalar Java Development Kit[1] (JDK): openJDK versión 1.8.0.
- Instalar Java Runtime Environment[2] (JRE): JRE versión 1.8.0.
- Instalación de Apache Ant[3]: Apache Ant versión 1.10.
- Descargarse el archivo ZIP de este proyecto.

I.2. Aplicación BucketServer

En este apartado vamos a explicar los pasos a realizar:

1. Descargar el fichero con extensión «.zip» del proyecto y descomprimirlo. Una vez lo tenemos descomprimido, accedemos al fichero bucket-server/build.xml. En este fichero se encuentra las instrucciones para la fase de compilación de las dependencias del proyecto.
2. Accedemos al fichero y comprobamos, que la estructura de directorios y ficheros se corresponde con la nuestra.
3. Procedemos a la ejecución del fichero build.xml para la compilación del proyecto utilizando el comando:

```
ant bucket-server/build.xml
```

Realizamos esta compilación para comprobar que todo funciona correctamente.

[1] Java Development Kit - Es un software que provee herramientas de desarrollo para la creación de programas en Java: `https://www.oracle.com/technetwork/es/java/javase/downloads/index.html`

[2] Java Runtime Environment - Es un conjunto de utilidades que permiten la ejecución de programas Java: `https://www.oracle.com/technetwork/es/java/javase/downloads/index.html`

[3] Apache Ant - Es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción: `https://ant.apache.org/`

4. Una vez la compilación funciona correctamente, tendremos que configurar varios archivos del proyecto.

   a) Accedemos al fichero bucket-server/src/bucket/server/BucketServer.java, dentro del fichero nos ubicamos en las líneas 91 a 94 del código. Sustituimos las cadenas "/TFG/bucket-server/gui" y "file:///var/www/html/TFG/bucket-server/gui/"por la ubicación actual de nuestro proyecto

   b) Accedemos al fichero bucket-server/conf/app.properties, dentro del fichero cambiamos la configuración del archivo. En la variable app.path, ponemos la ruta actual de nuestro proyecto y seguimos el mismo proceso para las variables app.external.uri y app.external.uri.

   c) Accedemos al fichero de configuración para la conexión a la base de datos, se encuentra en bucket-server/conf/META-INF/persistence.xml. Dentro del fichero modificaremos los valores "javax.persistence.jdbc.user" y "javax.persistence.jdbc.password"dependiendo de la configuración que hagamos en la base de datos. Este punto será explicado con más detalle en la **Sección Base de datos**.

5. Una vez hemos realizado la modificación de los ficheros, volvemos a utilizar el archivo build.xml para la compilación del proyecto. La razón es para que los cambios que hemos realizado en los ficheros tengan efecto. El comando que usamos es:

   ```
   ant bucket-server/build.xml
   ```

6. Ahora realizamos la instalación y configuración del servidor web *Apache HTTP Server*. Este punto se explica en detalle en la **Sección Servidor Apache**.

7. Para poner en marcha el servidor *Java* de *BucketServer*, necesitamos ir al directorio bucket-server/build/bucket-server/bin. Una vez dentro del directorio ejecutamos los siguientes comandos:

   ```
   sudo sh install-bucket.sh
   sudo sh start-bucket.sh
   ```

   Para parar el servidor de *BucketServer* o desinstalarlo, utilizamos los comandos:

   ```
   sudo sh stop-bucket.sh
   sudo sh uninstall-bucket.sh
   ```

8. Una vez tenemos en funcionamiento todas las partes, accedemos a la *URL* que hayamos definido en el servidor apache, en nuestro caso la *URL* es "https://localhost/TFG/bucket-server/gui/admin.html".

9. Finalmente ya podremos trabajar con la aplicación *BucketServer*.

I.3. Base de datos

Los pasos para la instalación y configuración de la base de datos se detalla a continuación:

1. Instalación de *MySQL Server*:

```
sudo apt update
sudo apt install mysql-server
sudo mysql_secure_installation
```

2. Creamos una base de datos en *MySQL*:

```
create database database\_name;
```

3. Ejecutamos el *script* del proyecto bucket-server/conf/db/create_bucketServer_database.sql dentro de la base de datos creada *database_name*. Para ejecutar el *script* dentro de *MySQL* usamos el comando:

```
source bucket-server/conf/db/create\_bucketServer\_database.sql
```

4. Creamos un usuario con contraseña en *MySQL*. Este usuario se pondrá en el archivo de configuración persistence.xml para la aplicación *BucketServer* y así podrá acceder a la base de datos.

```
create user user@localhost identified by 'password';
grant all privileges on databaseName. to user@localhost;
```

5. Una vez realizados todos estos pasos hemos terminado con la instalación y configuración de la base de datos *MySQL*.

I.4. Servidor Apache

Los pasos para la instalación y configuración del servidor Apache se detalla a continuación:

1. Instalación de Apache HTTP Server:

```
sudo apt update
sudo apt install apache2
```

2. Configuramos el cortafuegos, para que se garantice el acceso externo a los puertos web por defecto:

```
sudo ufw allow 'Apache'
sudo ufw allow 'Apache Secure'
sudo ufw allow 'Apache Full'
sudo ufw allow 'openSSH'
```

3. Verificamos que la instalación se ha realizado correctamente:

```
sudo systemctl status apache2
```

4. Ahora vamos a configurar una conexión segura *SSL* y un *proxy reverse*, para conectar el servidor Apache con el servidor *BucketServer*.

I.4.1. Configuración SSL Apache

Es necesario configurar una conexión con el protocolo *SSL*, para tener acceso a la cámara del dispositivo y tener una conexión segura.

1. Generamos un certificado con la herramienta *OpenSSL*[4], utilizamos el comando:

```
sudo openssl req -x509 -nodes -days 1095 -newkey rsa:2048
-out /etc/apache2/ssl/server.crt -keyout /etc/apache2/ssl/server.key
```

2. Activamos el módulo *SSL* de *Apache* mod_ssl [16] con el comando:

```
sudo a2enmod ssl
```

3. Creamos un enlace simbólico en el directorio apache default-ssl:

```
sudo ln -s /etc/apache2/sites-available/default-ssl.conf
/etc/apache2/sites-enabled/000-default-ssl.conf
```

4. Editamos el fichero 000-default-ssl.conf y añadimos dos líneas dentro del fichero:

```
sudo vim /etc/apache2/sites-enabled/000-default-ssl.conf
SSLCertificateFile    /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
```

5. Por último reiniciamos el servidor *Apache*:

```
sudo systemctl restart apache2
```

[4]OpenSSL es un proyecto software libre, tiene un paquete de herramientas de administración y bibliotecas relacionadas con la criptografía: `https://www.openssl.org/`

I.4.2. Configuración proxy reverse Apache

Nuestro servidor *Apache HTTP Server* no va a alojar los datos él mismo. En su lugar el contenido de los datos se obtendrá del servidor *backend*, en este caso se trata del servidor *Java* de la aplicación *BucketServer*. Cuando el servidor *Apache* recibe una petición de un cliente, se hace un *proxy* de esta petición al servidor *backend*, que genera el contenido y lo envía de vuelta al servidor *Apache*. Finalmente este genera la respuesta HTTP que irá de vuelta al cliente [11]. Un ejemplo de implementación típica es la Figura **I.1**.

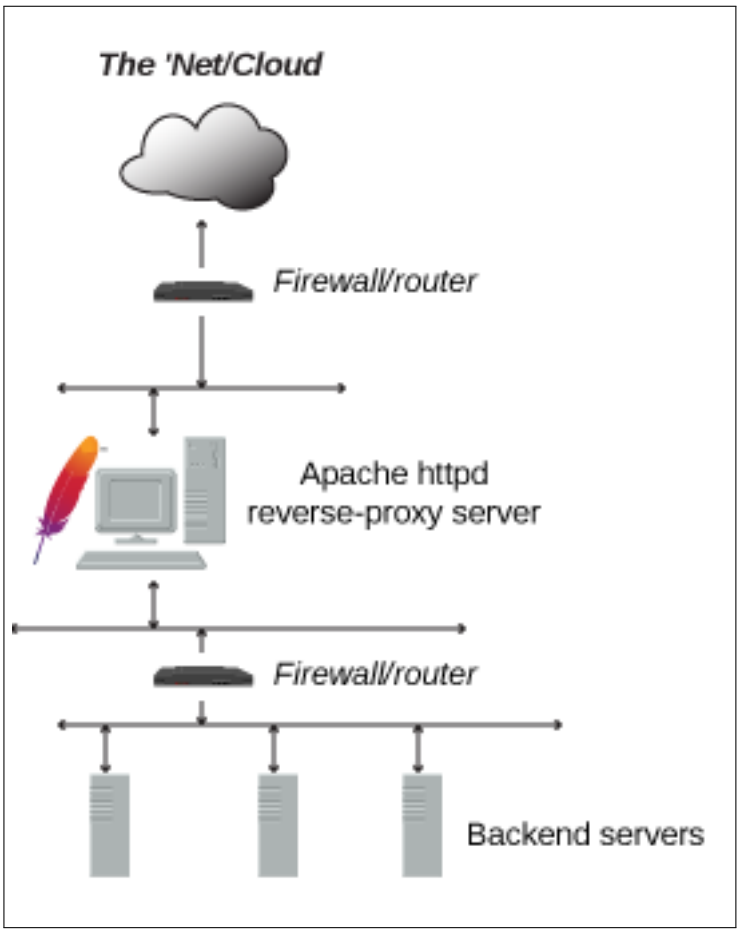


Figura I.1: Implementación de un Proxy Reverse [11]

1. Instalamos los módulos que vamos a necesitar:

```
sudo a2enmod proxy
sudo a2enmod prox_balancer
sudo a2enmod proxy_connect
sudo a2enmod proxy_html
sudo a2enmod proxy_http
```

2. Reiniciamos el servidor *Apache*:

```
sudo systemctl restart apache2
```

3. Modificamos el archivo /etc/apache2/sites-enabled/000-default.conf con el siguiente comando:

```
sudo vim /etc/apache2/sites-enabled/000-default.conf
```

4. El fichero queda de la siguiente forma; ver Figura **I.2**

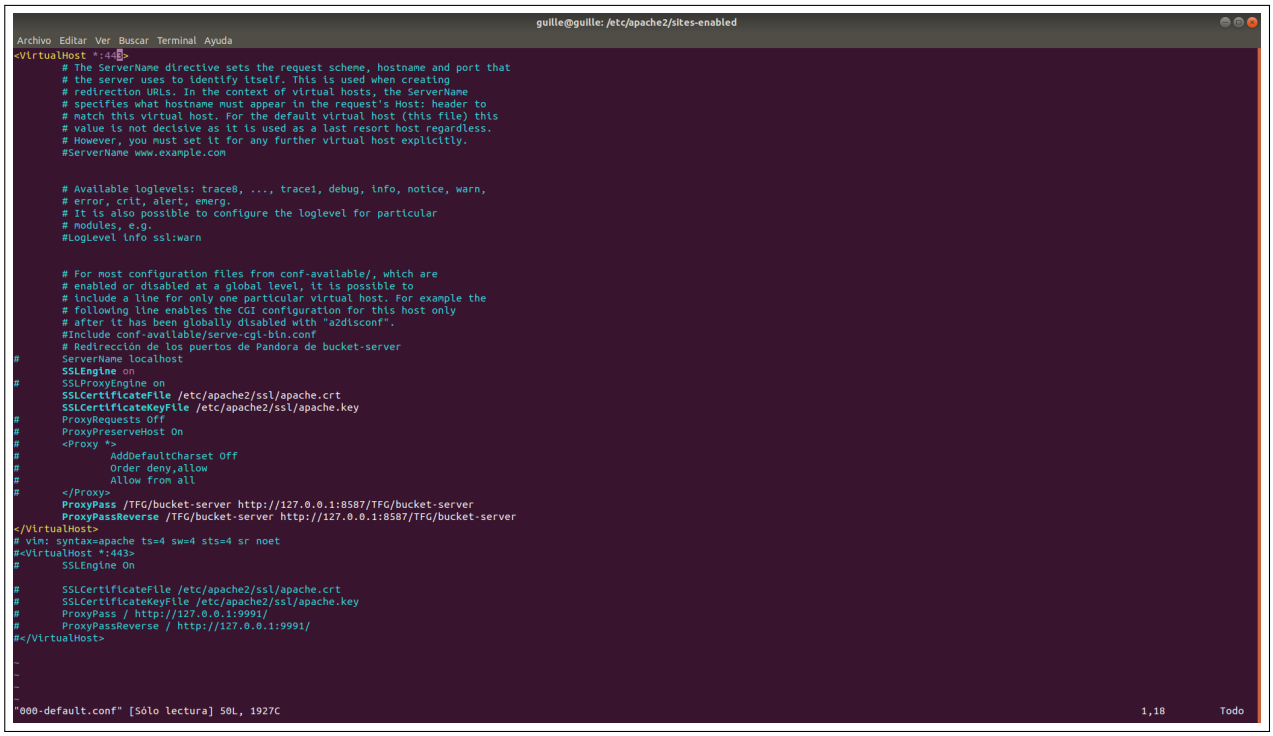


Figura I.2: Configuración del archivo 000-default.conf

5. Por último, volvemos a reiniciar el servidor *Apache*.

```
sudo systemctl restart apache2
```

Una vez hemos realizado todos estos pasos, hemos terminado con la instalación y configuración del servidor *Apache*.

Anexo II

Manual de usuario

Para acceder al sistema el primer paso consiste en introducir la URL de la aplicación en el navegador. Esta URL es de la forma https://dominio/bucket-server/gui/admin.html. Ver Figura **II.1**.

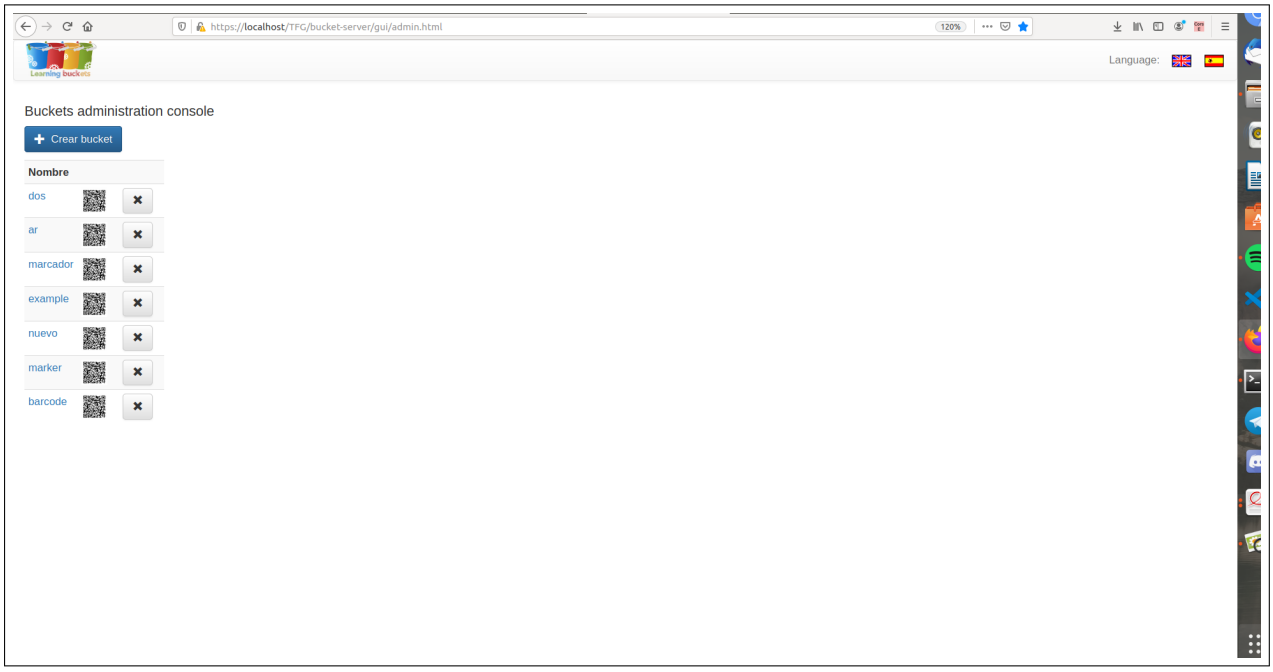


Figura II.1: Captura de la página para acceder y crear *buckets*

Una vez en la página del sistema puedes acceder a uno de los *buckets* ya creados o crea uno desde cero, haciendo click en el botón «Crear bucket».

El panel que se mostrará es el de la Figura **II.2**.

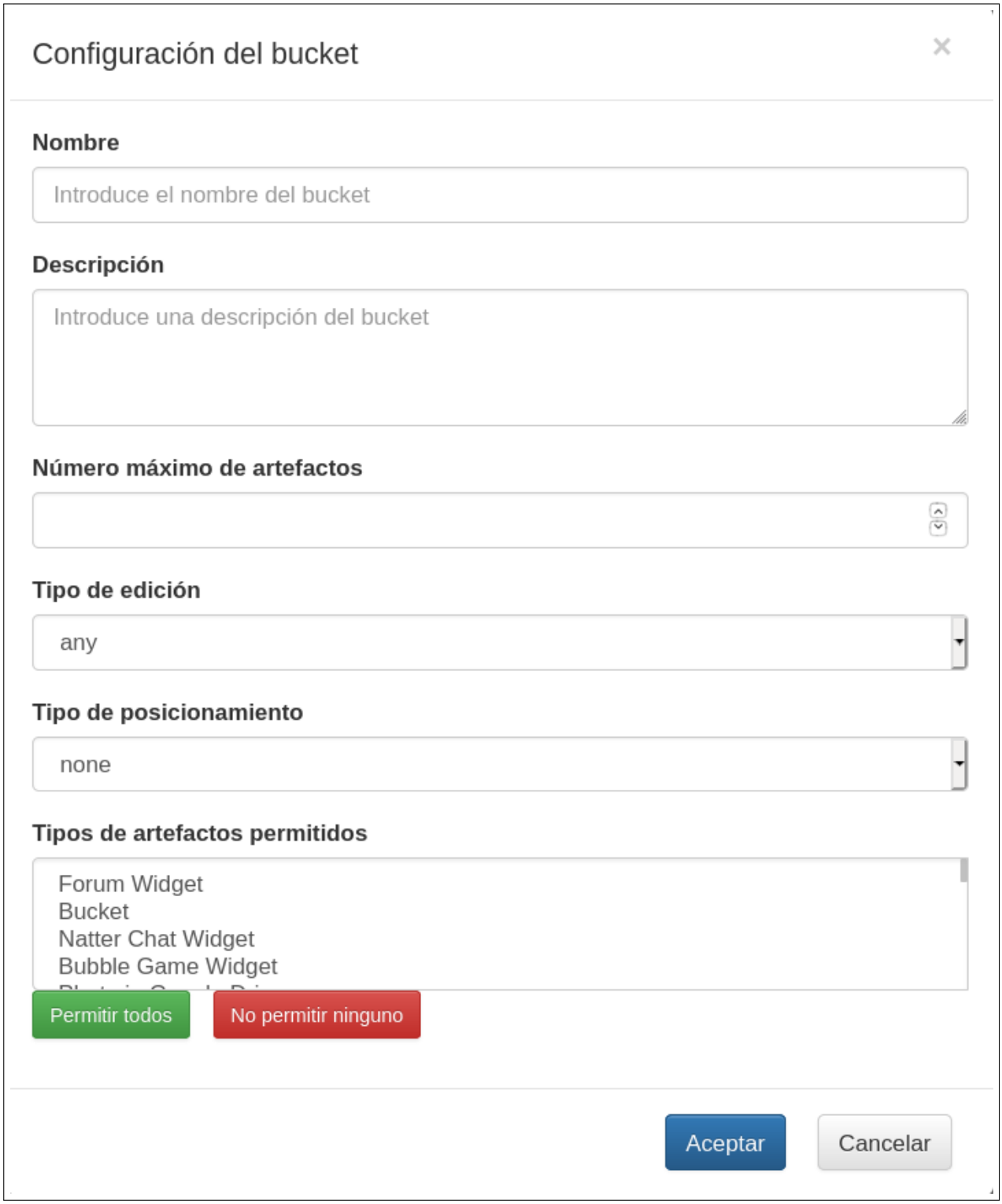


Figura II.2: Panel para la creación de un *bucket*

Si accedemos dento de un *bucket* la vista que se tendrá es la de la Figura **II.3**.

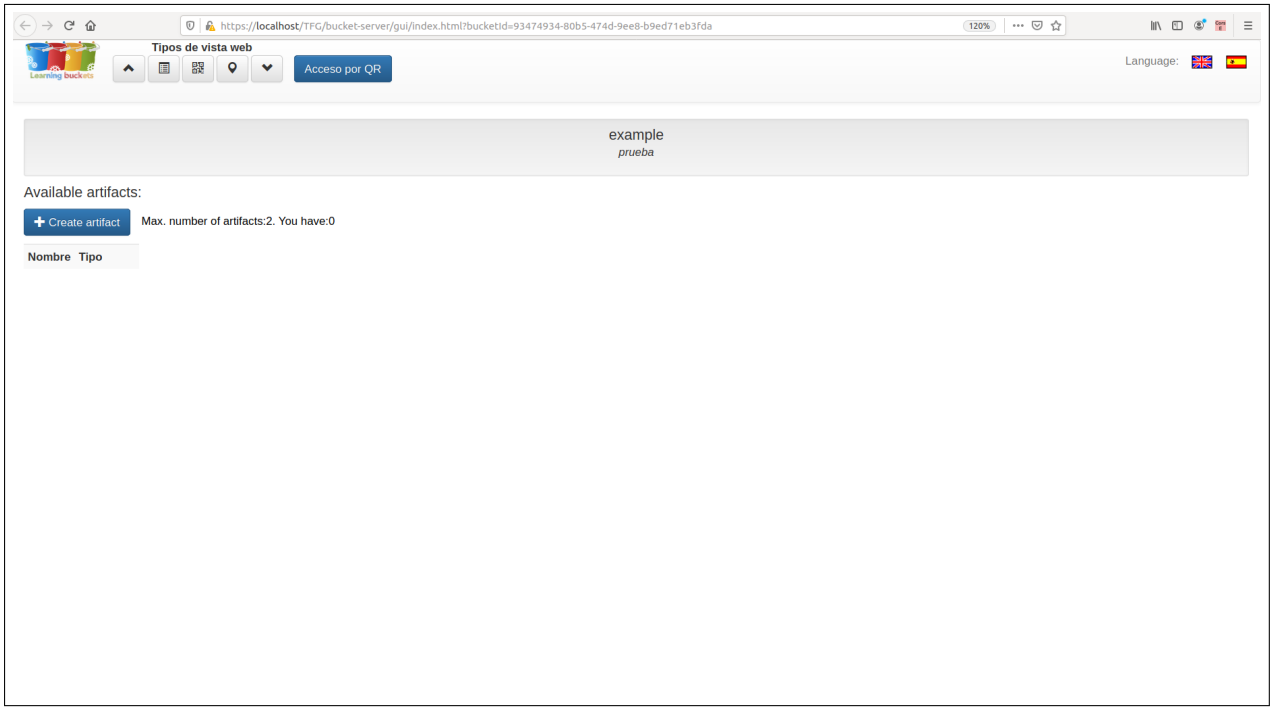


Figura II.3: Página web de un *bucket*

Si queremos cambiar el tipo de vista, para poner la vista de los marcadores artefactos o visualizar el contenido de los artefactos, tenemos que hacer click sobre los iconos de los tipos de vista web. También podemos seleccionar el botón acceso por QR para poder escanear los códigos QR del bucket para acceder al bucket o del modo AR para accceder al modo AR. Si queremos crear un artefacto tenemos que hacer click en el botón «Create artifact». El panel que se mostrará es el de la Figura **II.4**.

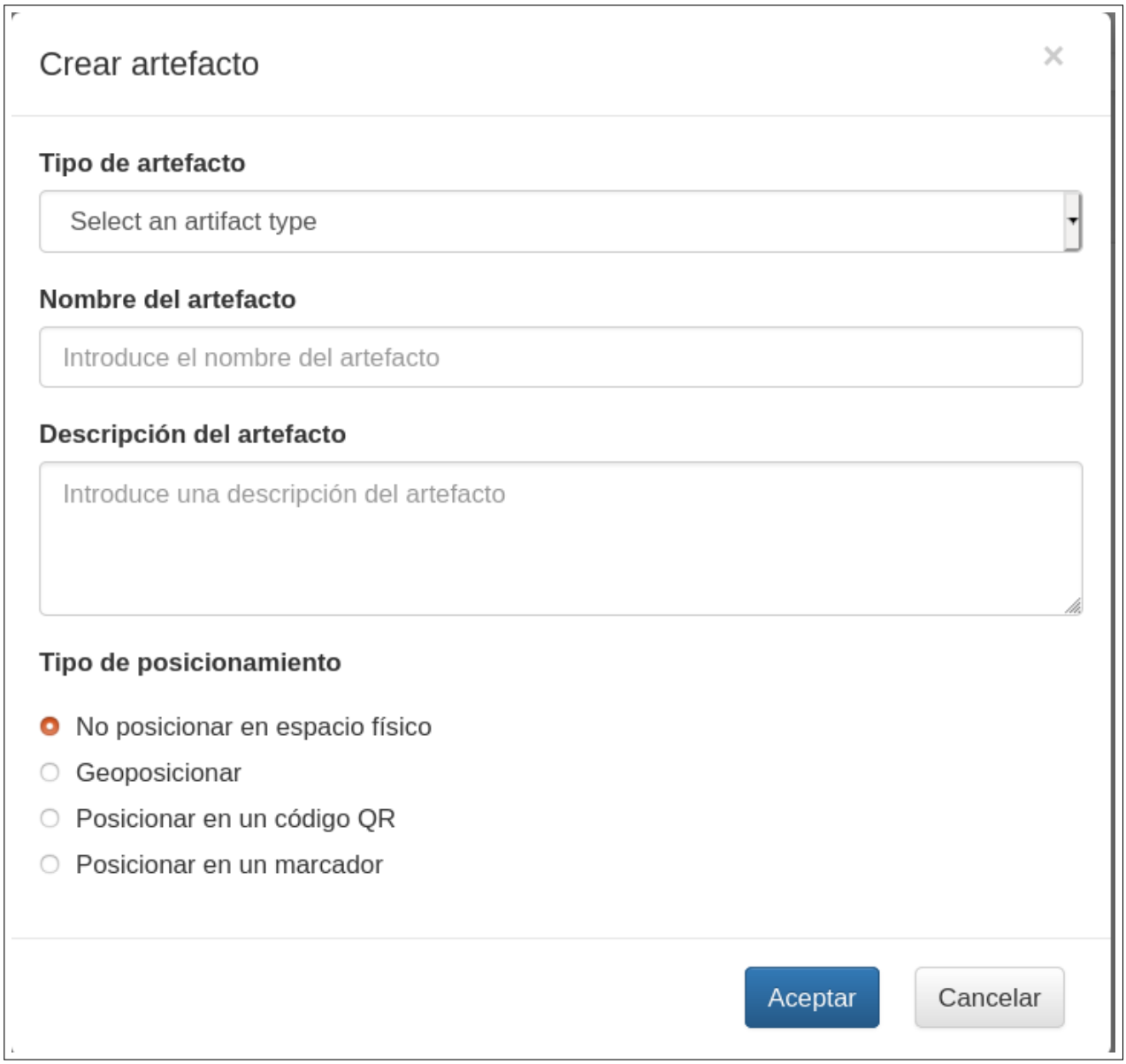


Figura II.4: Panel para la creación de un artefacto

Para poder acceder al modo AR es necesario que haya creado algún artefacto con el posicionamiento marcador o de geoposición. Si es así, en la vista de la Figura **II.3** aparecerá a mayores un código QR dentro del botón acceso por QR, como el de la Figura **II.5** y un botón *switch* para cambiar entre el modo web y modo ar como el de la Figura **II.6**.

Figura II.5: Ejemplo de código QR para acceder al modo AR y al bucket

Figura II.6: Vista dde un bucket con el interruptor switch para cambiar entre el modo AR y modo Web

Dentro del modo de realidad aumentada la vista que veremos cambiara dependiendo de los artefacto que se hayan creado. Si solo existen artefactos geoposicionados o en marcadores *barcode*, solo mostrara un *switch* para cambiar entre el modo web y ar, si existen los dos tipos de artefactos aparecerá otro *switch* para cambiar entre el reconocimiento de geoposición o marcadores *barcode* y si estamos reconociendo marcadores *barcode* y encontramos uno aparecerá un botón para acceder al contenido del marcador *barcode*. Un ejemplo de la vista que se mostraría es el de la Figura **II.7**

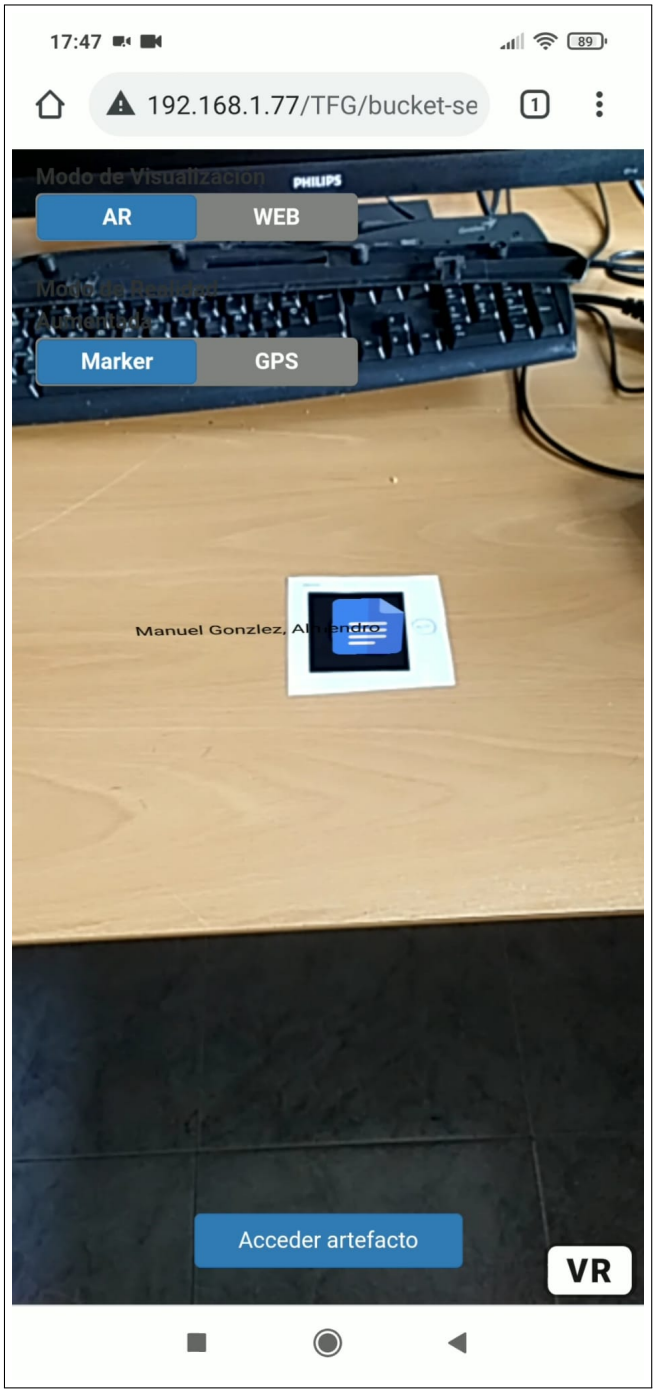


Figura II.7: Vista del modo AR con un marcador *barcode*

Anexo III

Enlaces de descarga

Los ficheros del código fuente de la aplicación *BucketServer* y de la documentación en *LaTeX* se puede del siguiente enlace de *Google Drive* `https://drive.google.com/drive/folders/1Oo5EfTiycKApHtzsqXRMniU-68GoHfZq?usp=sharing`