Universidad de las Ciencias Informáticas

Facultad 4

Título: “Plataforma para la gestión de contenidos de Realidad Aumentada a través de servicios en línea”.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Sadiel Lázaro Caballero Ramos

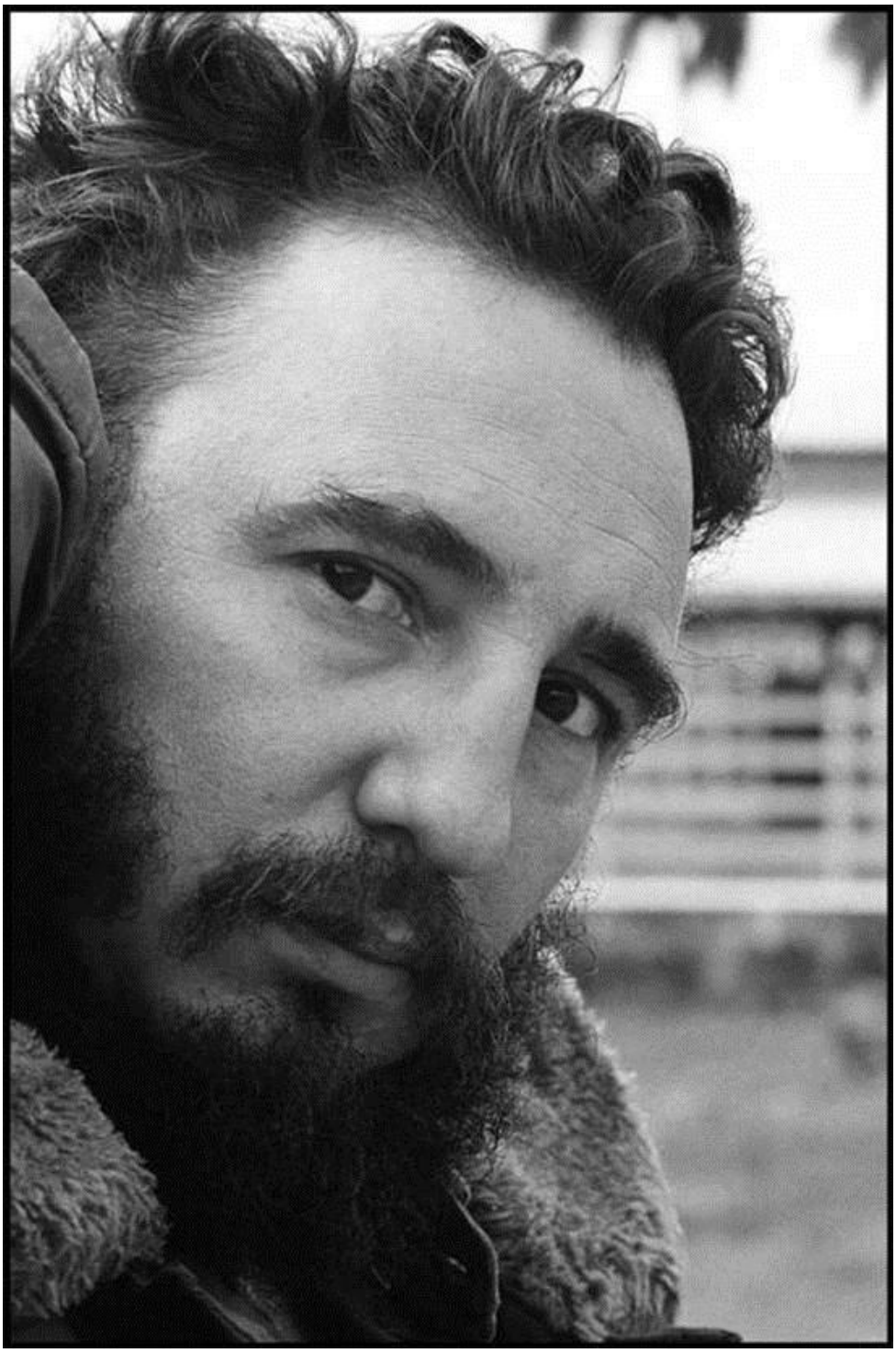
Tutores: MSc. Ernesto de la Cruz Guevara Ramírez

Ing. Alejandro Ravelo Julian

“Año 59 de la Revolución”

La Habana, Cuba

Junio 2017



“Una revolución solo puede ser hija de la cultura y de las ideas”.

Fidel Castro Ruz

Declaración de autoría

Declaramos ser los únicos autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Sadiel Lázaro Caballero Ramos

Autor

_____ _____

MSc. Ernesto de la Cruz Guevara Ramírez

Tutor

Ing. Alejandro Ravelo Julian

Tutor

Datos de Contactos:

MSc. Ernesto de la Cruz Guevara Ramírez.

Email: elguevara@uci.cu

- Graduado de Ingeniero en Ciencias Informáticas en la UCI, Cuba.
- Categoría docente: Asistente.
- 10 años de experiencia de trabajo con Gráficos por computadoras (Juegos, Visualización Médica y Científica).
- 7 años de experiencia de trabajo con Visión por computadoras, Realidad Aumentada.
- Master en Informática Aplicada.

Ing. Alejandro Ravelo Julian

Email: aravelo@uci.cu

- Graduado de Ingeniero en Ciencias Informáticas en la UCI, Cuba.
- 3 años de experiencia de trabajo con Gráficos por computadoras (Juegos, Visualización Médica y Científica).
- 3 años de experiencia de trabajo con Visión por computadoras, Realidad Aumentada.

Resumen

El centro Vertex de la facultad 4 de la Universidad de las Ciencias Informáticas cuenta con una línea de investigación de Realidad Aumentada. Las aplicaciones de Realidad Aumentada desarrolladas en dicho centro ocupan un espacio de almacenamiento excesivo con respecto a las aplicaciones que existen en el mercado. Esto sucede porque los recursos necesarios para el funcionamiento de la aplicación van incluidos dentro de su archivo de instalación. Para la solución del problema se planteó como objetivo general, desarrollar una plataforma para la gestión de contenidos de Realidad Aumentada que provea los contenidos en línea. Se propuso una solución basada en un servicio en la nube que pueda ser consumido por las aplicaciones de Realidad Aumentada para visualizar los elementos virtuales que se requieran, estos son imágenes, videos o elementos 3D. La solución permite crear aplicaciones con menor espacio de almacenamiento, debido a que los recursos necesarios para el correcto funcionamiento de las aplicaciones son distribuidos a través de un servicio en la nube. El correcto funcionamiento de la solución tiene como precondición que exista disponibilidad de los contenidos en la red. Con el desarrollo de este trabajo se logró disminuir el espacio de almacenamiento que ocupan las aplicaciones de Realidad Aumentada gracias al consumo de los recursos en línea. La solución ofrece una ventaja en ambientes donde existen conexiones de red, debido a que se pueden ofrecer contenidos de forma dinámica sin cambiar la aplicación.

Palabras claves: computación en la nube, informática, realidad aumentada, servicio en línea.

Índice

Índice de Figuras

Índice de Tablas

Índice de Gráficos

Introducción

La evolución de las tecnologías ha propiciado que el mundo esté en constante desarrollo. La informática es una de las ciencias que más ha evolucionado en los últimos tiempos.

Una de las tendencias actuales en la informática es la computación en la nube (*cloud computing*), la cual es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet. En este tipo de computación todo lo que puede ofrecer un sistema informático es un servicio, de modo que los usuarios puedan acceder a los servicios disponibles "en la nube de Internet" sin conocimientos (o, al menos sin ser expertos) en la gestión de los recursos que consume.

El almacenamiento en la nube se ha expandido al uso cotidiano. Un ejemplo universalmente utilizado y gratuito son las aplicaciones Google. El gigante de internet ha creado una *suite* ofimática que permite crear y almacenar documentos y archivos en la nube de forma fácil y flexible. Las cuales estarán disponibles en cualquier momento y desde cualquier lugar con sólo conectarse a Internet. Permite utilizar un total de 15 GB a compartir entre Google Drive, Gmail y Google Fotos. Dropbox (1) ofrece 2 GB de almacenamiento, totalmente gratuitos, compatible con los sistemas operativos más extendidos.

OneDrive es el servicio de Microsoft de almacenamiento en la nube. Al igual que Dropbox, dispone de una aplicación para *Smartphone* y tabletas de manera que siempre es posible acceder a los datos. Su capacidad es de 7 GB ampliables hasta 15 de forma gratuita.

A nivel empresarial y de negocios, probablemente sea más recomendable acudir a empresas de pago, como Amazon Web Services (2), Google Cloud Platform (3) y Windows Azure (4). Dependiendo de las necesidades de la empresa, los precios varían, por lo que es aconsejable estudiar distintas ofertas antes de decantarse por una en concreto (5).

En Cuba esta tecnología ha comenzado su camino hacia el desarrollo, en el nodo nacional de la red de datos del sistema de centros docentes del Ministerio de Educación Superior, se construyó una nube de servidores. La solución implementada fue desplegada sobre el sistema Xen Cloud Platform, caracterizada por su sencillez y por su robustez. A partir de la implementación de la solución anterior, se logró poner en servicio el portal de la editorial universitaria y establecer el procedimiento para su gestión, actualización y seguridad informática. En estos momentos se cuenta con dos sitios, uno con acceso desde Internet, con los contenidos de la Biblioteca Virtual de la EcuRed, por otra parte, el repositorio de

recursos educativos abiertos, disponible para los centros adscritos al Ministerio de Educación Superior de la República de Cuba (MES). Las experiencias adquiridas pueden ser válidas para los nodos de las redes de datos universitarias (6). La Universidad de las Ciencias Informáticas (UCI), no se ajena a esta tecnología y brinda servicios en la nube para el consumo de sus usuarios.

Otra de las tecnologías que se encuentra en auge en la actualidad es la Realidad Aumentada (7), término que se utiliza para definir un sistema de interacción que toma como entrada la información que proviene del mundo real y genera información de salida (como objetos, imágenes y texto) que se superpone en el tiempo real sobre la percepción que el usuario tiene del mundo real. Una de las últimas aplicaciones que evidencian el uso de la Realidad Aumentada es el videojuego Pokémon GO (8).

El centro Vertex, Entornos Interactivos 3D, de la facultad 4 de la UCI, ha desarrollado varios videojuegos en colaboración con el Instituto Cubano del Arte y la Industria Cinematográfica (ICAIC). Como producto agregado de cada videojuego se desarrolla una aplicación de Realidad Aumentada. Además, las aplicaciones necesitan en ocasiones mostrar más de un recurso asociado y esto hace que ocupen mayor espacio de almacenamiento. Como ejemplo, la aplicación de Realidad Aumentada para el libro interactivo de PostgreSQL llegó a tener un espacio de almacenamiento superior a los 600 MB. Lo cual es excesivo para este tipo de aplicaciones.

Teniendo en cuenta las posibilidades técnicas y el desarrollo alcanzado en el centro Vertex en las líneas de Videojuegos y Realidad Aumentada, se propone integrar el desarrollo de las aplicaciones de Realidad Aumentada con la computación en la nube.

El centro Vertex cuenta con una línea de investigación de Realidad Aumentada. Empleando el motor de desarrollo para la creación de juegos y contenidos 3D interactivos Unity3D. El desarrollo de estas aplicaciones está limitado a la utilización de marcadores para la creación de contenido de Realidad Aumentada, el flujo de dichas aplicaciones se limita a cargar la cámara del dispositivo, reconocer un marcador y enriquecer la realidad con elementos virtuales previamente empaquetados en la aplicación. Dichas aplicaciones ocupan un espacio grande al exportar ya que junto a la aplicación final van incluidos todos los recursos necesarios para su funcionamiento.

En la actualidad las aplicaciones para móviles tienden a desacoplar los contenidos y proponen cargar los recursos necesarios para el correcto funcionamiento de la aplicación desde Internet. Esto propicia que el tamaño de las aplicaciones disminuya al no incorporar recursos que posteriormente puede cargar desde un repositorio u otro almacenamiento directo a Internet. Dotando a las aplicaciones de mayor flexibilidad, al establecerlas como un reproductor de escenas de Realidad Aumentada. Para crear estas aplicaciones Unity3D cuenta con un tipo de archivo que se puede descargar y consumir por demanda de una aplicación, llamado *assetbundle*.

Con lo expuesto anteriormente se plantea el siguiente **problema de investigación**: ¿Cómo reducir el espacio en disco de las aplicaciones de Realidad Aumentada y cambiar dinámicamente sus contenidos?

La investigación tiene como **objeto de estudio** los servicios en la nube, constituyendo el **campo de acción:** los servicios en la nube para aplicaciones de Realidad Aumentada.

Para darle solución a este problema se propone el siguiente **objetivo general**: Desarrollar una plataforma para la gestión de contenidos de Realidad Aumentada que provea los contenidos en línea.

Para resolver el problema científico y dar cumplimiento al objetivo, se plantean las siguientes **tareas de investigación**:

1. Establecer los referentes teórico-metodológicos sobre las aplicaciones de Realidad Aumentada.
2. Seleccionar las tecnologías necesarias para brindar los servicios de *assetbundles*.
3. Selección de los algoritmos para la creación y visualización de *assetbundles*.
4. Implementar un componente de software para la creación de *assetbundles*.
5. Implementar un componente de software para la carga de *assetbundles*.
6. Implementar una plataforma web para la gestión de los *assetbundles* almacenados en la nube
7. Implementar una aplicación de Realidad Aumentada usando *assetbundles*.

Para el proceso de investigación y elaboración de este trabajo se tomará en cuenta la utilización de varios **métodos científicos de investigación** como:

- **Histórico-lógico:** Dicho método permitirá conocer todo lo referente a las tecnologías de la computación en la nube y Realidad Aumentada, además de brindar los conceptos y vocabularios propios del campo que permitan el entendimiento del trabajo.
- **Analítico-sintético:** Mediante el uso de este método se podrá estudiar los documentos que hacen referencia a la computación en la nube y a la Realidad Aumentada, para extraer los elementos más importantes de cada uno de ellos.
- **Revisión bibliográfica:** Este método permitirá determinar el estado del arte del objeto de investigación.
- **Matemáticos-Estadísticos:** Este método es utilizado para cuantificar los resultados de las pruebas realizadas al sistema. En la etapa de planificación se utilizó en el cálculo de las unidades de tiempo necesarias para el desarrollo del sistema.

El presente trabajo de diploma está estructurado de la siguiente forma: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones y referencias bibliográficas.

Capítulo #1. Fundamentación teórica

1.1 Introducción

En el presente capítulo se exponen los elementos teóricos fundamentales acerca de los servicios en línea, esencialmente de la computación en la nube y de su aplicación en la gestión de contenidos para aplicaciones de Realidad Aumentada. Se exponen los principales elementos que componen la computación en la nube, como los servicios e interfaces web, y se destacan sus características esenciales. Se identifica la arquitectura en la que se encuentra basada la computación en la nube, haciendo énfasis en las capas que conforma dicha arquitectura. Además, se exponen las principales características de los a*ssetbundles* y se caracterizan los lenguajes de programación y las herramientas empleadas en la solución del problema.

1.2 Conceptos y términos asociados al dominio de la investigación

En el presente epígrafe se muestran los conceptos que fueron identificados en el transcurso de la investigación para lograr un mejor entendimiento de los temas abordados.

1.3 Servicios en Línea

Los servicios que se brindan en línea están relacionados con las funciones de información, comunicación e interacción. Algunos de los principales servicios disponibles en línea son el correo electrónico, las conversaciones en línea (*chats*), acceso remoto a otros ordenadores y la computación en la nube (9).

1.3.1 Computación en la Nube

Existen varias definiciones para el término computación en nube (*Cloud Computing*). Según el Instituto Nacional de Normas y Tecnología (NIST: *National Institute of Standards and Technology*) la computación en la nube es un modelo que permite, convenientemente, el acceso bajo demanda a redes ubicuas para compartir un conjunto configurable de recursos de computación (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden proveer y liberar rápidamente con un mínimo esfuerzo de administración o interacción del proveedor del servicio. Este modelo de nube está compuesto por cinco características esenciales, tres modelos de servicios y cuatro modelos de despliegue (10).

También se puede definir computación en la nube como una tecnología que permite tanto a usuarios individuales como empresas, almacenar archivos y programas de forma remota, en lugar de utilizar discos duros y servidores (11). A los archivos almacenados en la nube se puede acceder por diferentes medios, como son el empleo de los servicios web (*web service*) e interfaces de usuario (interfaz web). La información almacenada en la nube puede variar desde música, videos, fotos hasta documentos, *assetbundles* y aplicaciones.

1.4 Servicio Web

Un Servicio Web es un conjunto de protocolos y estándares que se utilizan para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

De una manera más clara se podría definir que un servicio web es una función que diferentes servicios o equipos utilizan; es decir, solo se envían parámetros al servidor (lugar donde está alojado el servicio web) y éste responderá la petición. Entre algunas de las ventajas de los servicios web se destacan las siguientes:

- Aportan interoperabilidad entre aplicaciones de software, independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto, no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

La principal ventaja de utilizar un servicio web es que son prácticos debido a que son independientes de las aplicaciones (12)**.**

En los últimos años se ha popularizado un estilo de arquitectura software conocido como REST (*Representational State Transfer*). Este estilo ha supuesto una nueva colección de principios para el diseño de arquitecturas en red.

1.4.1 API REST

API REST es un estilo de arquitectura que permite con pocas operaciones manejar muchos recursos, es sencillo de construir y de adaptar y el consumo de recursos es escaso (13). A continuación, se exponen las principales características de los componentes que emplea una API REST para el intercambio de información entre aplicaciones.

API

Una API es que es la abreviatura de *Application Programming Interface*, o Interfaz de Programación de Aplicaciones*.* Una API consigue que los desarrolladores interactúen con los datos de la aplicación de un modo planificado y ordenado. Por ejemplo, la API de Facebook ofrece a los desarrolladores la posibilidad de obtener y mostrar todos los amigos de un usuario. Como las APIs incluyen documentación (una serie de instrucciones detalladas), es fácil para los desarrolladores obtener los datos que necesiten (14).

JSON

JSON (*JavaScript Object Notation)*, es un formato que permite la clasificación e intercambio de datos, principalmente utilizado en bases de datos (14).

Arquitectura REST

REST, abreviatura de *Representational State Transfer*, o Transferencia de Estado Representacional, es un estilo de arquitectura para diseñar aplicaciones en red.

La idea que transciende tras una REST, es ofrecer una alternativa sencilla para tecnologías complejas como SOAP. Para conseguirlo REST utiliza HTTP (14).

Peticiones HTTP

La REST API es mucho más efectiva gracias a HTTP (*Hyper Text Transfer Protocol*). Esto se debe a que permite compartir información entre un cliente (portátil, teléfono móvil o tableta) y un servidor (14)**.**

Un ejemplo sería una actualización de estado en Facebook. En nombre de un usuario de Facebook se envía una petición HTTP desde un código JavaScript (cliente) a la API de Facebook (servidor). Una vez que el usuario publica una actualización de estado el cliente (el código JavaScript) recibirá una respuesta del servidor de Facebook estableciendo que la operación se ha completado con éxito. Ver Figura 1 (14).

Un escenario de petición HTTP funciona de la siguiente manera:

1. Un cliente envía una petición HTTP a un servidor.
2. El servidor devuelve una respuesta HTTP.

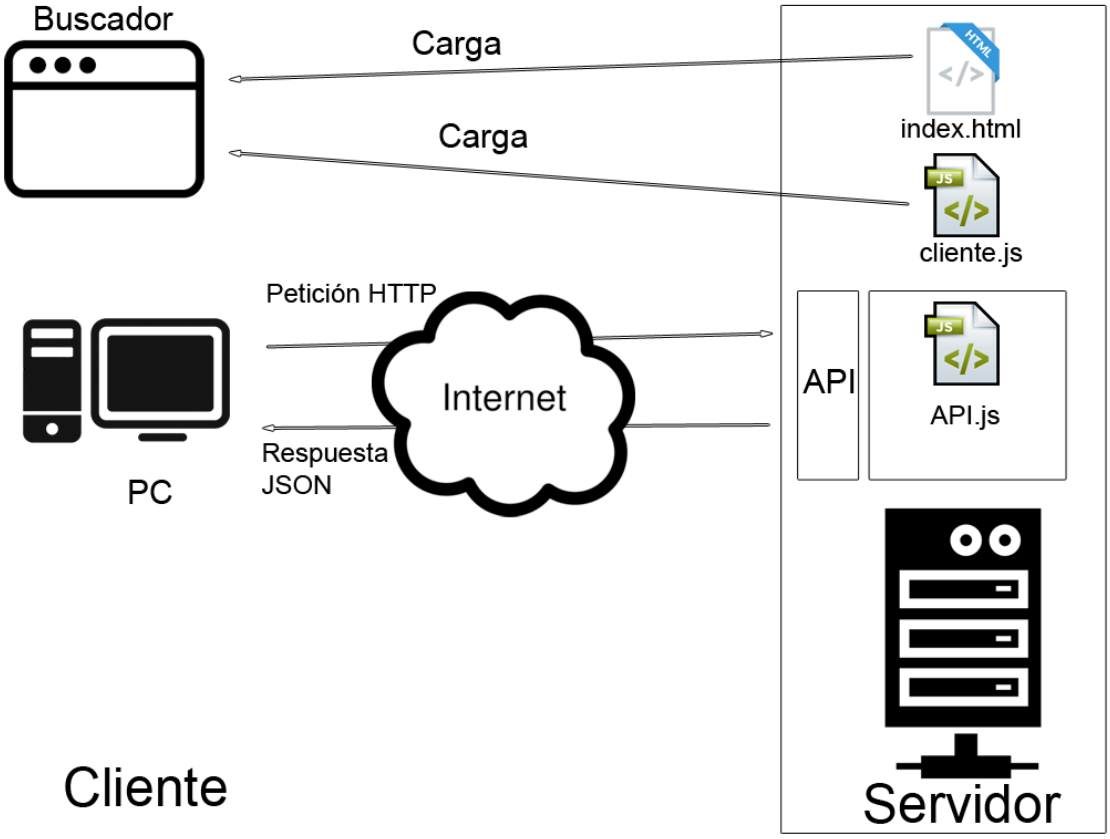


Figura 1 Funcionamiento de una API REST

1.5 Interfaz Web

Cuando se habla de sitios web, se denomina interfaz al conjunto de elementos de la pantalla que permiten al usuario realizar acciones sobre el sitio web que está visitando. Por lo mismo, se considera parte de la interfaz a sus elementos de identificación, de navegación, de contenidos y de acción.

En este sentido, es importante considerar que Jakob Nielsen (15), uno de los autores más citados en cuanto a la usabilidad de los sitios web, destaca que los elementos más importantes de la portada de todo sitio web se pueden resumir en cuatro postulados generales:

1. **Dejar claro el propósito del sitio:** se refiere a que el sitio debe explicar a quién pertenece y qué permite hacer a quienes lo visitan; se entiende que debe hacerlo de manera simple y rápida. Por ejemplo, ayuda en este sentido el cumplimiento de las normas referidas a uso de *URLs* (Localizador Uniforme de Recursos) y logotipos oficiales.
2. **Ayudar a los usuarios a encontrar lo que necesitan:** implica que debe contar con un sistema de navegación visible, pero que además deberá estar complementado por algún sistema de búsqueda que sea efectivo para acceder al contenido al que no se logra acceder o que no se encuentra a simple vista.
3. **Demostrar el contenido del sitio:** significa que el contenido se debe mostrar de manera clara, con títulos comprensibles por parte del usuario y con enlaces hacia las secciones más usadas que estén disponibles donde el usuario los busque. Ayudará en este sentido tener un seguimiento de las visitas para comprender qué es lo más visto y lo más buscado del sitio web.
4. **Usar diseño visual para mejorar y no para definir la interacción del sitio web:** se refiere a que los elementos gráficos del sitio web deben estar preparados para ayudar en los objetivos del sitio y no sólo como adornos utilizados para rellenar espacio. Aunque se trata de uno de los temas más debatibles, su alcance no es el de restringir el uso de imágenes y elementos gráficos, sino a que su uso sea adecuado para la experiencia de uso que se desea ofrecer (16).

1.6 Modelos de la Computación en la Nube

Según la definición del Instituto Nacional de Normas y Tecnologías (NIST), la computación en nube es un paradigma en evolución. El modelo de computación en nube del NIST destaca las cinco características esenciales de la misma proporcionando una línea base para discutir lo que es la computación en nube y un medio para comparar los servicios en la nube y sus estrategias de implementación (10).

1.6.1 Características esenciales de la Computación en la Nube

Según la definición del NIST (10), este modelo debe contar con cinco características esenciales:

1. **Autoservicio bajo demanda**

Un consumidor puede proveerse unilateralmente recursos de computación, tales como tiempo de servidor y almacenamiento en red, a medida que lo necesite, sin requerir interacción humana con el proveedor del servicio (10).

2. **Amplio acceso a la red**

Las capacidades están disponibles en la red y se acceden a través de mecanismos estándares que promueven el uso heterogéneo de plataformas de cliente ligeras o pesadas (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles o estaciones de trabajo) (10).

3. **Agrupamiento de recursos**

Los recursos de computación del proveedor están agrupados para servir a múltiples consumidores utilizando un modelo multidistribuido, con diferentes recursos físicos y virtuales asignados y reasignados dinámicamente de acuerdo a la demanda del consumidor. Existe una sensación de independencia de la posición, de manera que el cliente, generalmente no tiene el control o el conocimiento sobre la ubicación exacta de los recursos proporcionados, pero podría especificar una ubicación en un nivel más alto de abstracción (por ejemplo, país, estado o centro de datos). Ejemplos de recursos incluyen almacenamiento, procesadores, memoria y ancho de banda (10).

4. **Elasticidad rápida**

Las funcionalidades se pueden proporcionar de manera rápida y elástica y, en algunos casos, automáticamente. Sus características de aprovisionamiento dan la sensación y pueden adquirirse en cualquier cantidad o momento (10).

5. **Servicio medido**

Los sistemas en nube controlan y optimizan automáticamente el uso de recursos, potenciando la capacidad de medición en un nivel de abstracción apropiado al tipo de servicio (10).

1.7 Arquitectura de Computación en la Nube

La computación en la nube basa su arquitectura haciendo una separación entre hardware, plataforma y aplicaciones. Quedando las capas software como servicio, plataforma como servicio e infraestructura como servicio (17).

1.7.1 Software como Servicio

Se encuentra en la capa más alta y consiste en la entrega de aplicaciones completas como un servicio. El proveedor de tecnología de información y comunicación (TIC) ofrece el software como servicio (SaaS). Para ello dispone de una aplicación que se encarga de operar y mantener y que frecuentemente es desarrollada por el proveedor. Con ella se encarga de dar servicio a multitud de clientes a través de la red, sin que estos tengan que instalar ningún software adicional (17).

1.7.2 Plataforma como Servicio

En orden descendente, la plataforma como servicio (PaaS) es la siguiente capa. Básicamente su objetivo se centra en un modelo en el que se proporciona un servicio de plataforma con todo lo necesario para dar soporte al ciclo de planteamiento, desarrollo y puesta en marcha de aplicaciones y servicios web a través de ella. El proveedor es el encargado de escalar los recursos en caso de que la aplicación lo requiera, así como del rendimiento óptimo y seguridad de acceso. Para desarrollar aplicaciones se necesitan bases de datos, herramientas de desarrollo y en ocasiones servidores y redes. Con PaaS, el cliente únicamente se enfoca en desarrollar, depurar y probar, puesto que la herramienta necesaria para el desarrollo de software es ofrecida a través de Internet, lo que teóricamente permite aumentar la productividad de los equipos de desarrollo

Con el uso de PaaS, se abstrae del hardware físico al cliente, lo cual es interesante para muchos desarrolladores web, y es probable que llegue a reemplazar a las empresas de alojamiento tradicionales (17).

1.7.3 Infraestructura como Servicio

Infraestructura como servicio (IaaS) corresponde a la capa más baja. La idea básica es la de hacer uso externo de servidores para espacio en disco, base de datos, enrutadores, *swtiches,* así como tiempo de computo, evitando de esta manera tener un servidor local y toda la infraestructura necesaria para la conectividad y mantenimiento dentro de una

organización. Con una IaaS, se tiene una solución en la que se paga por consumo de recursos solamente usados: espacio en disco utilizado, tiempo de CPU, espacio para base de datos y transferencia de datos (17).

1.8 Realidad Aumentada

La Realidad Aumentada es un sistema de interacción que toma como entrada la información que proviene del mundo real y genera información de salida (tal como objetos, imágenes y texto) que se superpone en el tiempo real sobre la percepción que el usuario tiene del mundo real. Se consigue un complemento virtual que enriquece la realidad en el conocimiento que el usuario tiene sobre los objetos de su entorno.

Resulta de la “unión” entre mundo real con el mundo virtual, donde el usuario de este tipo de aplicaciones, podrá ver (a través de una cámara, o de dispositivos especiales de visión) objetos generados por ordenador que se integran en el mundo real (7).

Para la creación de contenidos de Realidad Aumentada el hardware juega un papel importante, pero el software es el encargado de realizar todo el proceso de creación, de ahí que sea tan importante contar con la plataforma adecuada de acuerdo a cada necesidad. Por ejemplo, Vuforia (18) y Wikitude (19) se encuentran entre las mejores herramientas, también están ARToolKit (7) y EasyAR (20) que son alternativas que no son tan conocidas como las demás, pero que se pueden alcanzar resultados aceptables. Al diseñar una aplicación de Realidad Aumentada, el centro Vertex cuenta con la herramienta Unity3D.

1.8.1 Vuforia

Vuforia es un SDK (Kit de Desarrollo de Software) que permite construir aplicaciones basadas en la Realidad Aumentada, una aplicación desarrollada con Vuforia utiliza la pantalla del dispositivo como un "lente mágico" en donde se entrelazan elementos del mundo real con elementos virtuales (como letras e imágenes). Al igual que con Wikitude, la cámara muestra a través de la pantalla del dispositivo, vistas del mundo real, combinados con objetos virtuales como: modelos, bloque de textos e imágenes

Una aplicación desarrollada con Vuforia ofrece las siguientes funcionalidades:

- Reconocimiento de Texto.
- Reconocimiento de Imágenes.

- Rastreo robusto. (el marcador fijado no se perderá tan fácilmente incluso cuando el dispositivo se mueva, o sea parcialmente ocluido).
- Detección Rápida de los marcadores.
- Detección y rastreo simultáneo de marcadores (18).

1.8.2 Wikitude

Wikitude es una SDK de Realidad Aumentada disponible para dispositivos con los sistemas operativos iOS, Windows y Android. Con ella, gracias a las imágenes tomadas directamente desde los *smartphone*, se pueden añadir una capa de información donde ver restaurantes, cafeterías o incluso, seguidores de Twitter que se encuentran próximos (19).

1.8.3 ARToolKit

ARToolKit es un conjunto de biblioteca para C/C++ que sirven para la creación de aplicaciones de Realidad Aumentada. Para ello proporciona una serie de funciones para la captura de vídeo y para la búsqueda de ciertos patrones en las imágenes capturadas mediante técnicas de visión por computador. También proporciona una serie de ejemplos y utilidades de gran ayuda al programador que quiera realizar este tipo de aplicaciones (7).

1.8.4 EasyAR

EasyAR es un motor de Realidad Aumentada, soporta Realidad Aumentada basado en objetivos planos, una carga suave y reconocimiento para más de 1000 objetivos locales, reproducción de video, video transparente y *streaming* de video, admite reconocimiento de código QR, seguimiento multi-objetivo. EasyAR no muestra marcas de agua y no tiene límites de tiempo de reconocimiento (20).

1.8.5 Unity3D

Unity3D es un motor de desarrollo para la creación de juegos y contenidos 3D interactivos, con las características que es completamente integrado y que ofrece innumerables funcionalidades para facilitar el desarrollo de videojuegos. Unity3D nombra *asset* a sus recursos, dígase texturas, audios, modelos 2D o 3D. Y los comprime en archivos llamados *assetbundles*, archivos que solo Unity3D o juegos creados en Unity3D pueden leer. (21).

1.8.6 *Assetbundles*

Los *assetbundles* son archivos que se pueden exportar desde Unity3D, y que contienen *assets* de la preferencia del usuario. Estos archivos utilizan un formato de compresión propio y se pueden descargar por demanda por su aplicación.

Los *assetbundles* se diseñaron para simplificar la descarga del contenido a su aplicación. Los *assetbundles* pueden contener cualquier tipo de *asset* reconocido por Unity3D, según lo determine la extensión en el nombre de archivo.

Al trabajar con *assetbundles*, el flujo de trabajo típico es:

Durante el proceso de creación de una aplicación, el desarrollador prepara los *assetbundles* y los sube a un servidor.

- **Construir *assetbundles***. Los *assetbundles* se crean en el editor a partir de los *assets* en la escena.
- **Subir los *assetbundles* a un medio de almacenamiento externo**. Este paso no involucra al editor de Unity3D o algún otro de los canales de Unity3D, pero se coloca para mostrar el procedimiento completo.

En tiempo de ejecución, la aplicación cargará los *assetbundles* bajo demanda y empleará los *assets* individuales en cada *assetbundles* conforme se necesite.

- **Descargar los *assetbundles* en tiempo de ejecución desde tu aplicación**. Mediante el empleo de un *script* los *assetbundles* son cargados bajo demanda de una aplicación.
- **Cargar objetos a partir de *assetbundles***. Una vez el *assetbundle* es descargado, se puede acceder a cada *asset* contenido en el *bundle*

Una de sus principales ventajas de los *assetbundles* es que permiten reducir el tamaño inicial de cualquier aplicación (22).

1.9 Lenguajes de Programación

1.9.1 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un

estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes (23).

1.9.2 HTML

HTML son las siglas designadas para "*Hyper Text Markup Language*", que traducido al español significa "Lenguaje de Marcas de Hipertexto". HTML es un lenguaje utilizado en la informática, cuyo fin es el desarrollo de las páginas web, indicando cuales son los elementos que la componen, orientando hacia cuál será su estructura y también su contenido, básicamente es su definición; por medio del HTML se indica tanto el texto como las imágenes pertenecientes a cada página de internet (24).

1.9.3 PHP

PHP son las siglas en inglés de **"***Hypertext Pre-Processor***"** que al traducirlo al español significa **"**Lenguaje de Programación Interpretado**"**. Este lenguaje es al que se le debe la visualización de contenido dinámico en las páginas web. Todo el código PHP es invisible para el usuario, porque todas las interacciones que se desarrollan en este lenguaje son por completo transformadas para que se puedan ver imágenes, variedad de multimedia y los formatos con los que somos capaces de interactuar añadiendo o descargando información de ellos (25)**.**

1.10 Metodología para el desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software (26). Se aplican estas metodologías con el objetivo asegurar la eficiencia en el proceso de producción de software.

1.10.1 Selección de la metodología de desarrollo de software

Para guiar el ciclo de vida del proyecto en desarrollo se seleccionó la metodología programación extrema o *Extreme Programming* (XP). Debido a que XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software y promoviendo el trabajo en equipo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los

cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Además, XP tiene como objetivos fundamentales establecer las mejores prácticas de Ingeniería de Software en los desarrollos de proyectos, mejorar la productividad de los proyectos y garantizar la calidad del software desarrollando, haciendo que este supere las expectativas del cliente (27).

1.11 Herramientas

En los siguientes epígrafes se exponen las herramientas que son empleadas en el desarrollo del proyecto, así como sus principales características y funciones.

1.11.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (28).

1.11.2 Yii 2

Yii es un *framework* de PHP de alto rendimiento, basado en componentes para desarrollar aplicaciones web modernas en poco tiempo. El nombre Yii significa "simple y evolutivo" en chino. También se puede considerar como un acrónimo de *Yes ItIs* (que en inglés significa *Sí, eso es*).

Actualmente existen dos versiones principales de Yii: la versión 1.1 y la versión 2.0. Para la versión 1.1, que es de la generación anterior, actualmente solo se ofrece mantenimiento. La versión 2.0 está completamente reescrita y adopta las últimas tecnologías y protocolos, incluidos *Composer*, PSR, *namespaces*, *traits*, etc. La versión 2.0 representa la actual generación del *framework* y su desarrollo recibirá el principal esfuerzo en los próximos años. Esta guía está basada principalmente en la versión 2.0 del *framework* (29).

1.11.3 WampServer

WampServer es un entorno de desarrollo web para Windows para la creación de aplicaciones web con Apache, PHP y bases de datos MySQL*database*. También incluye PHPMyAdmin y SQLiteManager para manejar tus bases de datos.

Provee a los desarrolladores con los cuatro elementos necesarios para un servidor web: un Sistema Operativo (Windows), un manejador de base de datos (MySQL), un software para servidor web (Apache) y un software de programación script Web (PHP (generalmente), Python o PERL), debiendo su nombre a dichas herramientas. Lo mejor de todo es que WAMP5 es completamente gratuito. WAMP incluye, además de las últimas versiones de Apache, PHP y MySQL, versiones anteriores de las mismas, para el caso de que se quiera testear en un entorno de desarrollo particular. El uso de WAMP permite servir páginas HTML a Internet, además de poder gestionar datos en ellas, al mismo tiempo WAMP, proporciona lenguajes de programación para desarrollar aplicaciones Web (30).

1.12 Conclusiones Parciales

Luego de analizados los temas abordados durante el presente capítulo, se arribó a las siguientes conclusiones:

- El estilo de arquitectura más adecuado a emplear en la solución al problema de investigación es API REST. Debido a que posibilita que el intercambio de datos entre las aplicaciones de la propuesta de solución sea más rápida y consuma menos recursos.

- La propuesta de solución enmarca su arquitectura empleando la capa más baja de la arquitectura de computación en la nube, infraestructura como servicio. Lo que posibilita al usuario la capacidad de alojar sus recursos en un servidor remoto brindándole procesamiento, memoria y espacio en disco.

Capítulo #2. Solución propuesta

2.1 Introducción

En el presente capítulo se expone todo lo referente a la solución propuesta al problema de investigación. Se detallan las historias de usuarios y a partir de estas se crea un plan de duración del proyecto contando además con las iteraciones por las que se van a dividir las historias de usuario, además, se especifican la arquitectura y los patrones de diseño a tener en cuenta para el funcionamiento de la solución.

2.2 Soluciones Técnicas

Para la solución al problema de investigación, se propone el desarrollo de un servicio en línea, específicamente usando computación en la nube. Este servicio estará a disposición del usuario, el cual va a ser capaz de gestionar sus aplicaciones de Realidad Aumentada con los archivos *assetbundles* alojados en la nube. También va a poder almacenar en la nube sus propios archivos para el posterior consumo de los usuarios. El servicio en línea cuenta con una sección donde todos usuarios podrán hacer uso de las herramientas necesarias tanto para construir los archivos *assetbundles* como para que las aplicaciones de Realidad Aumentada puedan consumir del servicio en la nube. Se propone también el desarrollo de un servicio web, con el objetivo de que los usuarios puedan explorar los recursos que brinda la plataforma web sin necesidad de acceder a ella.

2.3 Fase de Exploración

El ciclo de vida de la metodología XP comienza con la fase de exploración, en esta etapa se confeccionan las historias de usuarios (HU) que describen las funcionalidades que el cliente desea estén presentes en la aplicación; además, propone definir el alcance general del proyecto. Para la realizar la programación de esta fase es necesario un tiempo de pocas semanas.

2.3.1 Historias de Usuarios

Las Historias de Usuarios (HU) tienen el mismo propósito de los casos de uso, son escritas por los propios clientes y son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán al proceso de construcción de los *test* de aceptación (empleados para verificar que las HU han sido implementadas correctamente) (31).

Según Kent Beck cada HU recoge los siguientes aspectos:

Número: Número asignado a la HU.

Nombre de HU: Atributo que contiene el nombre de la HU.

Fecha: Fecha en la cual fue redactada la HU.

Usuario: El usuario del sistema que utiliza o protagoniza la HU.

Prioridad en el negocio: Contiene el nivel de prioridad de la HU en el negocio. Es Alta en caso de que la HU sea indispensable en el negocio, Media en caso de que su realización o no afecte el negocio y Baja cuando no se considera una prioridad.

Riesgo de desarrollo: Contiene el nivel de riesgo en caso de no realizarse la HU. Es Alta, si el riesgo de no realizar la HU incide en el funcionamiento de la plataforma, Media si el riesgo de no realizarla es medianamente importante, y Baja en caso de que no se considere un riesgo tardar en la realización de la HU y no incida en el funcionamiento de la plataforma.

Puntos estimados: Este atributo es una estimación hecha por el equipo de desarrollo sobre el tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo, y un día equivale a 0.2 puntos.

Iteración asignada: Especifica la iteración a la que pertenece la HU correspondiente.

Descripción: Posee una breve descripción de lo que realizará la HU.

Para asignar los valores de prioridad se tiene en cuenta lo siguiente:

Alta: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, las que el cliente define como principales para el control del sistema.

Media: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo.

Prototipo: Prototipo de la interfaz.

Historia de usuario	
Número: 1	Nombre de la HU: Crear Usuario

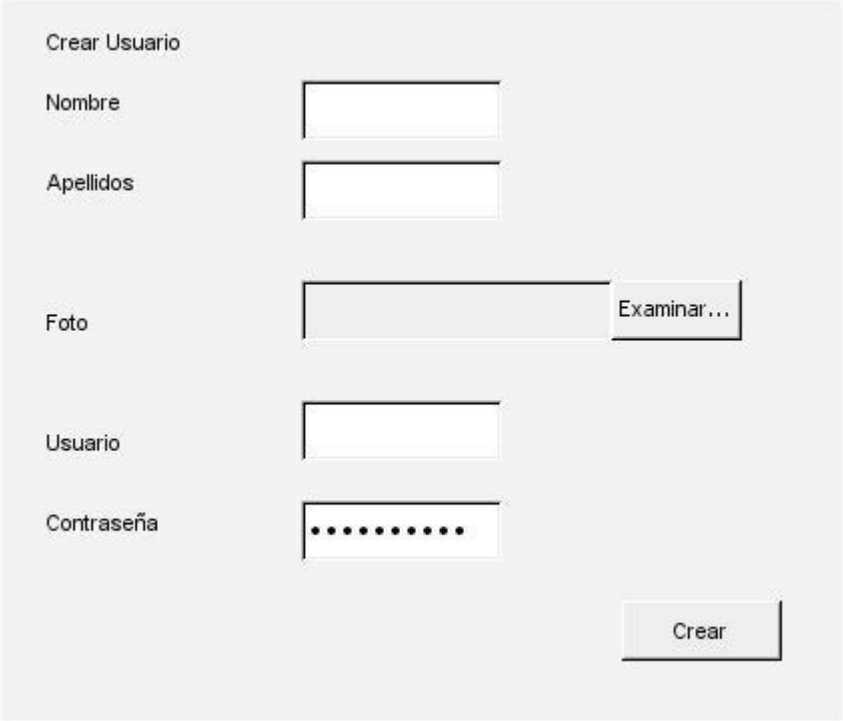
Fecha: 29.1.17	Usuario: Usuario
Prioridad de Negocio: Alta	Riesgo de Desarrollo: Alta
Puntos Estimados: 1	Iteración Asignada: 1
Programador Responsable: Sadiel Caballero Ramos	
Descripción**:** El usuario se podrá crear una cuenta de usuario en el sistema para acceder a la plataforma web.	
Prototipo:  	

Tabla 1. Historia de usuario # 1.

Historia de usuario	
Número: 5	Nombre de la HU: Crear *Assetbundles*
Fecha: 29.1.17	Usuario: Usuario

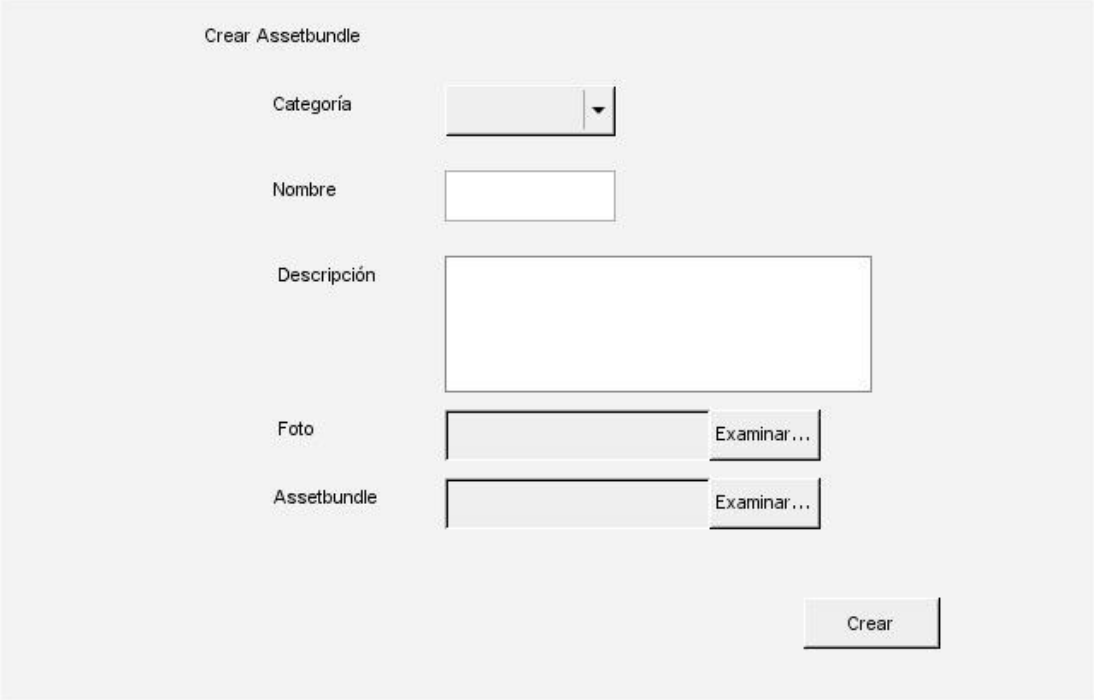
Prioridad de Negocio: Alta	Riesgo de Desarrollo: Alta
Puntos Estimados: 1	Iteración Asignada: 2
Programador Responsable: Sadiel Caballero Ramos	
Descripción**:** El sistema debe permitir al usuario almacenar sus *assetbundles* en la nube.	
Prototipo:  	

Tabla 2. Historia de usuario # 5.

Historia de usuario	
Número: 9	Nombre de la HU: Consultar *Assetbundles*
Fecha: 29.1.17	Usuario: Usuario
Prioridad de Negocio: Alta	Riesgo de Desarrollo: Alta
Puntos Estimados: 3.4	Iteración Asignada: 3

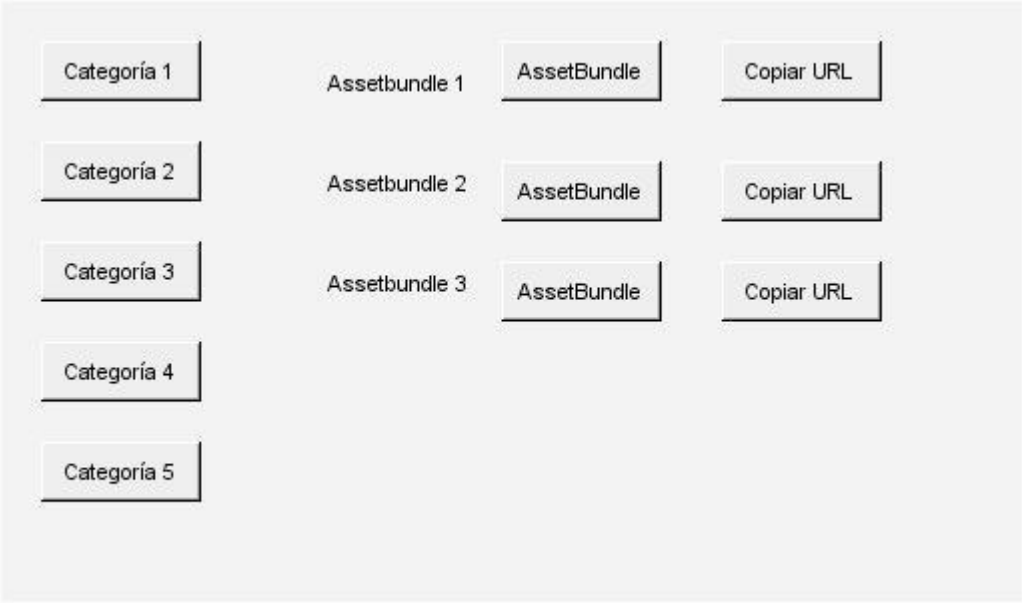
Programador Responsable: Sadiel Caballero Ramos
Descripción**:** El sistema debe ofrecer tanto la *URL* del *assetbundle* como una opción de descarga. Además de un servicio web para visualizar los *assetbundles* alojados en la nube.
Prototipo: 

Tabla 3. Historia de usuario # 9.

Historia de usuario	
Número: 10	Nombre de la HU: Crear Categorías.
Fecha: 29.1.17	Usuario: Administrador
Prioridad de Negocio: Alta	Riesgo de Desarrollo: Alta
Puntos Estimados: 1	Iteración Asignada: 4
Programador Responsable: Sadiel Caballero Ramos	

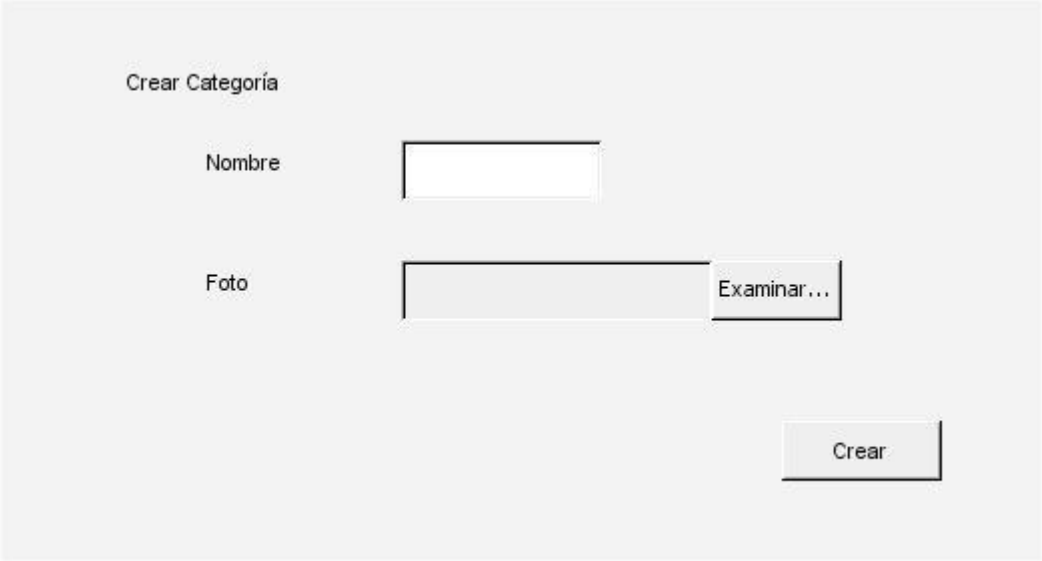
Descripción: El sistema debe permitir al administrador crear las categorías por las que se dividen los *assetbundles*.
Prototipo: 

Tabla 4. Historia de usuario # 10.

2.4 Fase de Planificación

En la fase de planificación es donde el cliente prioriza cada una de las HU y los programadores realizan una posible estimación del esfuerzo que será necesario para la programación de cada una de ellas. Se utiliza como medida los puntos de estimación para realizar las estimaciones de los esfuerzos que están asociados a la implementación de las HU, anteriormente priorizadas por los clientes. Las historias generalmente valen de 1 a 3 puntos. La planificación se puede realizar basándose en el tiempo o el alcance.

2.4.1 Plan de Estimación

A continuación, se muestra mediante una tabla la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

No.	Historia de Usuario	Prioridad	Esfuerzo Estimado
1	Crear Usuario	Alta	1

2	Modificar Usuario	Media	1
3	Eliminar Usuario	Media	0.8
4	Consultar Ayuda y Herramientas	Media	1
5	Crear *Assetbundles*	Alta	1
6	Modificar *Assetbundles*	Alta	1
7	Eliminar *Assetbundles*	Media	0.8
8	Mostrar *Assetbundles*	Alta	0.8
9	Consultar *Assetbundles*	Alta	3.4
10	Crear Categorías	Alta	1
11	Modificar Categorías	Media	1
12	Eliminar Categorías	Media	0.8
13	Mostrar Categorías	Media	0.8
14	Crear Comentario	Baja	1
15	Eliminar Comentario	Baja	0.8
16	Mostrar Comentario	Baja	0.8

Tabla 5. Estimación de esfuerzos.

2.4.2 Plan de Iteraciones

El plan de entrega cuenta con varias iteraciones que no deben exceder las tres semanas de desarrollo. Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se decide realizar el sistema en 5 iteraciones, las cuales se describen de manera más detallada a continuación:

Iteración 1

Se implementan las HU 1, 2, 3 y 4. Al finalizar la implementación de las tres primeras HU, el usuario será capaz de poder acceder al sistema utilizando un usuario y una contraseña, además de poder modificar o eliminar su cuenta. Luego de que concluya la implementación de la cuarta HU, el usuario poseerá las herramientas necesarias para crear y consumir *assetbundles* alojados en la nube a través del servicio web y los *scripts* que ponen a disposición del usuario. Además, de una sección de ayuda que sirva de guía al usuario a la hora de crear o consumir los *assetbundles.*

Iteración 2

Se implementan las HU 5, 6, 7, y 8. Al concluir la implementación de las HU 5, 6, 7 y 8 el usuario va a tener la capacidad de poner a disposición de los demás usuarios sus *assetbundles* a través de la plataforma web, también será capaz de modificar y eliminar solo los datos de los *assetbundles* cargados por el propio usuario.

Iteración 3

Al finalizar la implementación de la HU 9 el usuario podrá consumir los *assetbundles* alojados en la nube, ya sea a través del uso de las *URLs*, como de la funcionalidad de descargar el archivo, esta última opción se brinda a los usuarios que deseen tener los *assetbundles* en su ordenador. Además, el usuario dispondrá de un servicio web a través de Unity3D, para visualizar los *assetbundles* alojados en la nube.

Iteración 4

Se implementan las HU 10, 11, 12 y 13. Al concluir las implementaciones de las HU 10, 11, 12 y 13 el usuario, en este caso el administrador del sistema, va a poder crear, mostrar, modificar y eliminar las categorías por las que se dividirá los *assetbundle* almacenados en la nube.

Iteración 5

Se implementan las HU 14, 15 y 16. Al finalizar las implementaciones de las HU 14, 15 y 16 el usuario va a poder crear, mostrar y eliminar comentarios que realice acerca de los *assetbundles*.

2.4.3 Plan de duración de las iteraciones

La metodología XP es la encargada de guiar el ciclo de vida del proyecto en desarrollo, para ello se crea un plan de duración donde se expondrá el tiempo que necesitarán las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como objetivo fundamental mostrar la duración de cada una de las iteraciones en las que está dividida la fase de desarrollo del proyecto, así como el orden en que serán implementadas las HU en cada iteración según la prioridad asignada por el cliente.

Iteración	Historia de Usuario	Duración total de la iteración
Iteración 1	Crear Usuario	3.8
	Modificar Usuario	
	Eliminar Usuario	
	Consultar Ayuda y Herramientas	
Iteración 2	Crear *Assetbundles*	3.6
	Modificar *Assetbundles*	
	Eliminar *Assetbundles*	
	Mostrar *Assetbundles*	
Iteración 3	Consultar *Assetbundles*	3.4
Iteración 4	Crear Categorías	3.6
	Modificar Categorías	
	Eliminar Categorías	
	Mostrar Categorías	
Iteración 5	Crear Comentarios	2.6

	Eliminar Comentarios	
	Mostrar Comentarios	

Tabla 6. Plan de duración de las iteraciones.

2.4.4 Plan de Entrega

Historia de Usuario	Final de Iteración 1 (2 de marzo)	Final de Iteración 2 (28 de marzo)	Final de Iteración 3 (20 de abril)	Final de Iteración 4 (16 mayo)	Final de Iteración 5 (2 de junio)
Crear Usuario	V1.0				
Modificar Usuario	V1.0				
Eliminar Usuario	V1.0				
Consultar Ayuda y Herramientas	V1.0				
Crear *Assetbundles*		V1.1			
Modificar *Assetbundles*		V1.1			
Eliminar *Assetbundles*		V1.1			
Mostrar *Assetbundles*		V1.1			

Consultar *Assetbundles*			V1.2		
Crear Categorías				V1.3	
Modificar Categorías				V1.3	
Eliminar Categorías				V1.3	
Mostrar Categorías				V1.3	
Crear Comentarios					V1.4
Eliminar Comentarios					V1.4
Mostrar Comentarios					V1.4

Tabla 7. Plan de entrega.

Al realizar la estimación, el tiempo de desarrollo obtenido fue de 17 semanas. Los tiempos estimados de las HU variaron de 1 a 4 semanas.

2.5 Fase de Diseño

La arquitectura y diseño del software son procesos que proporcionan la estructura, funcionamiento e interacción entre las partes del software. A continuación, se presentan las tarjetas CRC, el patrón arquitectónico y los patrones de diseño que se utilizaron en el desarrollo del servicio en línea para la gestión de contenidos de Realidad Aumentada.

2.5.1 Tarjetas CRC

Las tarjetas CRC (clase, responsabilidad y colaboración) son una metodología para el diseño de software orientado por objetos creada por Kent Beck y Ward Cunningham, Son un puente de comunicación entre diferentes participantes. El mayor valor de las tarjetas CRC es que permite romper con el modo de procedimiento y de pensamiento para apreciar mejor la tecnología de objetos. También permiten que todo el equipo pueda contribuir al diseño del proyecto. Cuantas más personas puede ayudar a diseñar el sistema, mayor es el número de buenas ideas incorporadas (32).

Las tarjetas CRC están compuestas por los siguientes campos:

Clase: Nombre de la clase que se corresponde con la tarjeta.

Responsabilidad: Describe las funcionalidades que contiene clase.

Colaboración: Enuncia las clases que guardan relación con la que se describe en la tarjeta.

A continuación, aparecen cinco tarjetas CRC que corresponden a clases dentro del sistema:

Tarjeta CRC	
Clase: *CategoriesController*	
Responsabilidad:	Colaboración:
• Listar todos los modelos de Categoría. • Crear una nueva Categoría. • Modificar una Categoría existente. • Eliminar una Categoría existente.	*CategoriesForm* *Categories* *UploadFile*

Tabla 8. Tarjeta CRC de la clase *CategoriesController*

Tarjeta CRC	
Clase: *CommentController*	
Responsabilidad:	Colaboración:

• Listar todos los modelos de Comentarios. • Crear un Nuevo Comentario. • Eliminar un Comentario existente.	*Comment* *Controller*

Tabla 9. Tarjeta CRC de la clase *CommentController*

Tarjeta CRC	
Clase: *ElementController*	
Responsabilidad:	Colaboración:
• Listar todos los modelos de Elementos. • Crear un nuevo Elemento. • Modificar un Elemento existente. • Eliminar un Elemento existente.	*ElementForm* *Element* *UploadFile*

Tabla 10. Tarjeta CRC de la clase *ElementController*

Tarjeta CRC	
Clase: *SiteController*	
Responsabilidad:	Colaboración:
• Mostrar en pantalla el *homepage*. • Autenticar un usuario para acceder al sistema. • Desconectar un usuario del sistema.	*ResgisterForm* *Users* *ContactForm* *LoginForm*

• Registrar un usuario en el sistema. • Crear nuevos directorios para que se almacene información de cada usuario.	

Tabla 11. Tarjeta CRC de la clase *SiteController*

Tarjeta CRC	
Clase: *UsersController*	
Responsabilidad:	Colaboración:
• Listar todos los modelos de Usuarios. • Crear un nuevo Usuario. • Modificar un usuario existente. • Eliminar un usuario existente.	*ElementForm* *Users*

Tabla 12. Tarjeta CRC de la clase *UsersController*

2.5.2 Arquitectura del Sistema

La arquitectura Modelo-Vista-Controlador (MVC) es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos**,** Vistas y Controladores, o lo que es lo mismo, *Model, Views & Controllers*. MVC fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos *frameworks* de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

Las diversas partes o conceptos en los que se deben separar el código de la aplicación son las siguientes:

Modelos

Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos estarán habitualmente en una base de datos, por lo que en los modelos se encontrarán todas las funciones que accederán a las tablas.

Vistas

Las vistas contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá renderizar los estados de la aplicación en HTML. En las vistas solo estarán los códigos HTML y PHP que permite mostrar la salida**.**

En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a estos. Las vistas requerirán los datos a los modelos y ellas se generarán la salida, tal como la aplicación requiera.

Controladores

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento o una búsqueda de información. (33).

En la figura 2 se muestra las clases por las que están compuestas las distintas capas de la arquitectura MVC. En la capa Vista se encuentran las siguientes clases: *create*, *index*, *update* y *view*; todas ellas, para mayor organización, se guardan en directorios con nombre igual a la controladora a la que pertenezcan, esto ocurre en los casos de *categories*, *comment*, *element*, *site* y *users*. El directorio *categories* contiene una clase más, con nombre *all*. El directorio site contiene las siguientes clases: *about*, error, *contact*, *index*, *login* y *register*. La capa Vista también contiene la clase *main*, la cual se encuentra dentro del directorio *layouts*.

En la capa Controlador se encuentran las clases, CategoriesController, CommentController, ElementController, UsersController, SiteController.

En la Capa Modelo se encuentran las siguientes clases: Categories, CategoriesForm, Comment, CommentForm, Element, ElementForm, Users, LoginForm, RegisterForm, Like, y Rol.

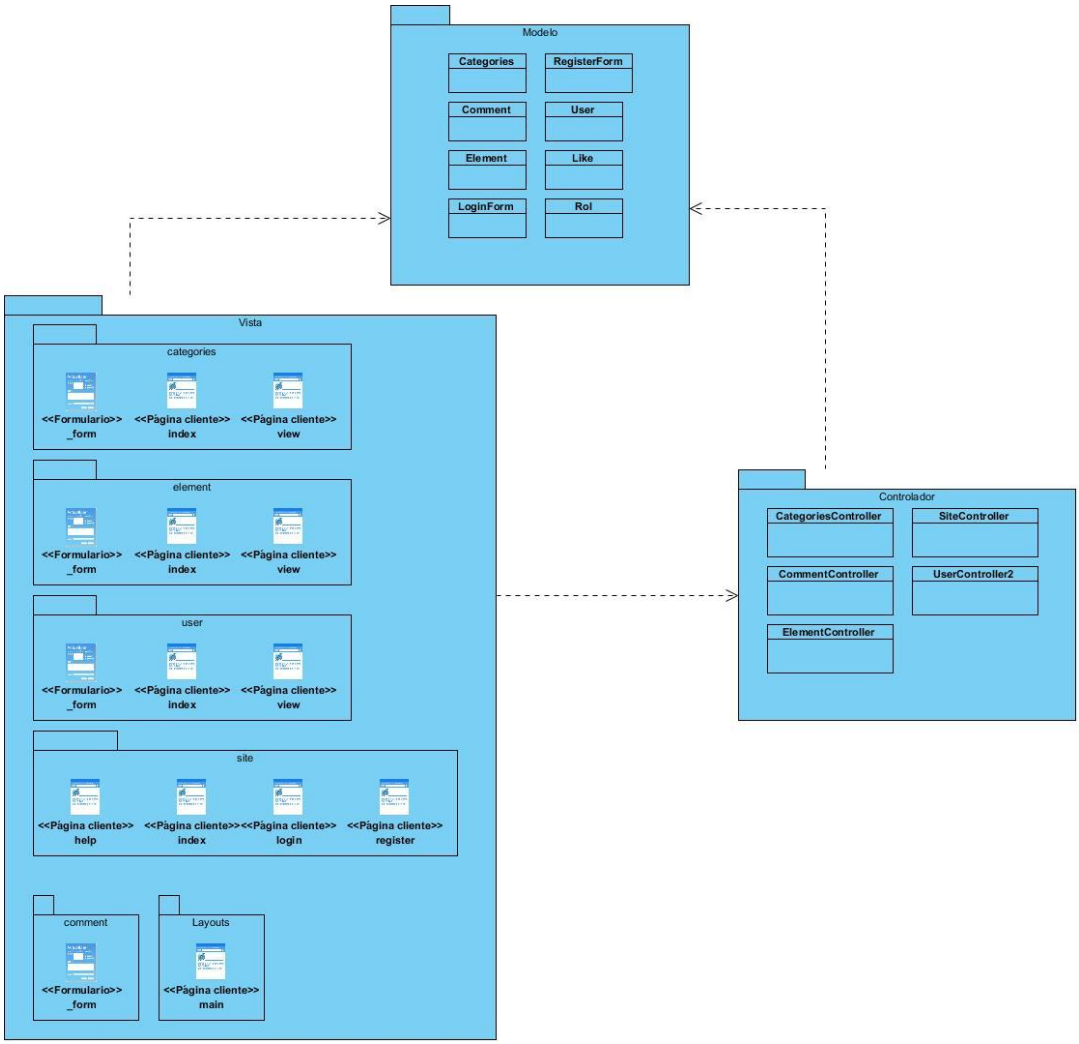


Figura 2. Arquitectura Modelo-Vista-Controlador del proyecto

2.6 Elementos del diseño

Para el diseño de la plataforma para la gestión de contenidos de Realidad Aumentada a través de servicios en línea, se hizo uso de los patrones *General Responsibility Assignment Software Patterns* (GRASP).

Patrón Bajo Acoplamiento y Alta Cohesión: Este patrón se encarga de que no existan dependencias innecesarias entre las clases y que realicen funciones estrechamente relacionadas con su contenido. El empleo de este patrón se evidencia ya que las clases del proyecto fueron diseñadas bajo estas especificaciones.

Patrón Controlador: Este patrón es utilizado en algunas clases del proyecto, una de ellas es la clase *CategoriesController*, la cual se encarga de recibir los datos del usuario y enviarlos a distintas clases según el método llamado. Este patrón sugiere dividir la lógica de negocios de la capa de presentación, para así aumentar la reutilización de código.

Patrón Creador: Este patrón es el encargado de la creación de instancias de objetos o clases ya que posee la información necesaria para este proceso, la utilización de este patrón se evidencia con la clase *CategoriesController*.

Patrón Experto: Este patrón es utilizado para asegurar que cada clase tenga todos los elementos necesarios para realizar sus responsabilidades.

Patrón de diseño GoF

Para el desarrollo de la plataforma se empleó también el patrón de diseño GoF, específicamente *Singleton*. El objetivo de este patrón es asegurarse de que de una clase solo existe una instancia y que esta es accesible, ofreciendo un punto de acceso a ella. La utilización de este patrón se evidencia en la clase *Connection*.

2.7 Conclusiones Parciales

Luego de analizados los temas abordados durante el presente capítulo, se arribó a las siguientes conclusiones:

- La arquitectura del sistema y los elementos de diseños utilizados proporcionan la estructura, funcionamiento e interacción entre las partes del software. Permitiendo que una parte del sistema varié de forma independiente sin afectar a las demás partes

- Las Historias de Usuario permiten acercar el lenguaje del usuario al desarrollador, de forma que se permita hablar un mismo lenguaje ambos y se eviten ambigüedades que puedan suponer pérdidas de tiempo notables.

Capítulo #3. Implementación y prueba del sistema

3.1 Introducción

En el presente capítulo se expone todo lo referente a la fase de implementación, la cual se emplea para lograr una mayor eficiencia en la implementación de las distintas HU. Además, Se realizan las pruebas del sistema para evaluar la calidad y eficacia del software.

3.2 Fase de Implementación

En esta fase se efectúa la implementación de todas las HU, donde se crean las tareas de programación para implementar con la mayor eficiencia todas las HU. A continuación, se detallan las tareas de ingeniería de cada HU.

Número	Historia de Usuario	Tareas por Historia de Usuario
1	Crear Usuario	1. Registrar Usuario 2. Autenticar Usuario
2	Modificar Usuario	1. Modificar Usuario
3	Eliminar Usuario	1. Eliminar Usuario
4	Consultar Ayuda y Herramientas	1. Consultar Ayuda 2. Consultar Herramientas
5	Crear *Assetbundles*	1. Crear *Assetbundles*
6	Modificar *Assetbundles*	1. Modificar *Assetbundles*
7	Eliminar *Assetbundles*	1. Eliminar *Assetbundles*
8	Mostrar *Assetbundles*	1. Mostrar *Assetbundles* 2. Visualizar *Likes* y Descargas

9	Consultar *Assetbundles*	1. Consumir *URL* del *Assetbundle* 2. Consumir Servicio Web
10	Crear Categorías	1. Crear Categorías
11	Modificar Categorías	1. Modificar Categorías
12	Eliminar Categorías	1. Eliminar Categorías
13	Mostrar Categorías	1. Mostrar Categorías
14	Crear Comentarios	1. Crear Comentarios
15	Eliminar Comentarios	1. Eliminar Comentarios
16	Mostrar Comentarios	1. Mostrar Comentarios

Tabla 13. Tareas por HU.

Tarea de Ingeniería	
Número de tarea: 1	Número de Historia de Usuario: 1
Nombre de la tarea: Registrar Usuario	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha Inicio: 6/2/17	Fecha Fin: 8/2/17
Descripción: Se encarga de registrar a un nuevo usuario en el sistema.	

Tabla 14. Tarea de Ingeniería 1 de la HU 1.

Tarea de Ingeniería	
Número de tarea: 2	Número de Historia de Usuario: 1

Nombre de la tarea: Autenticar Usuario	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 9/2/17	Fecha Fin: 10/2/17
Descripción: Se encarga de brindar al usuario la opción de acceder al sistema a través de la funcionalidad autenticar usuario.	

Tabla 15. Tarea de Ingeniería 2 de la HU 1.

Tarea de Ingeniería	
Número de tarea: 1	Número de Historia de Usuario: 5
Nombre de la tarea: Crear A*ssetbundles*	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 3/3/17	Fecha Fin:9/3/17
Descripción: El sistema debe permitir al usuario almacenar sus *assetbundles* en la nube	

Tabla 16. Tarea de Ingeniería 1 de la HU 5.

Tarea de Ingeniería	
Número de tarea: 1	Número de Historia de Usuario: 9
Nombre de la tarea: Consumir *URL* del A*ssetbundle*	
Tipo de tarea: Desarrollo	Puntos Estimados:0.4
Fecha Inicio: 29/3/17	Fecha Fin:30/3/17
Descripción: Se brinda al usuario un componente para copiar la dirección *URL* de los *assetbundles* alojados en la nube.	

Tabla 17. Tarea de Ingeniería 1 de la HU 9.

Tarea de Ingeniería	
Número de tarea: 2	Número de Historia de Usuario: 9
Nombre de la tarea: Consumir Servicio Web	
Tipo de tarea: Desarrollo	Puntos Estimados:3
Fecha Inicio: 31/3/17	Fecha Fin:20/4/17
Descripción: El sistema debe ofrecer un servicio web para visualizar los *assetbundles* alojados en la nube.	

Tabla 18. Tarea de Ingeniería 2 de la HU 9.

Tarea de Ingeniería	
Número de tarea: 1	Número de Historia de Usuario: 10
Nombre de la tarea: Crear Categorías	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 21/4/17	Fecha Fin: 27/4/17
Descripción: Se encarga de implementar una clase que permita el administrador del sistema crear las diferentes categorías por las que se dividirán los distintos *assetbundles* alojados en la nube.	

Tabla 19. Tarea de Ingeniería 1 de la HU 10.

3.2.1 Estructura y descripción de la aplicación

Luego de concluidas las fases de exploración, planificación y determinado el tiempo de desarrollo, el producto final queda estructurado de la siguiente manera (ver Figura 3):

Los usuarios podrán gestionar los recursos almacenados en el servidor (Nube) mediante el empleo de la interfaz web de la plataforma. Para visualizar y crear las aplicaciones los

usuarios dispondrán de una plataforma de edición y una de ejecución, mediante el empleo de una API REST.

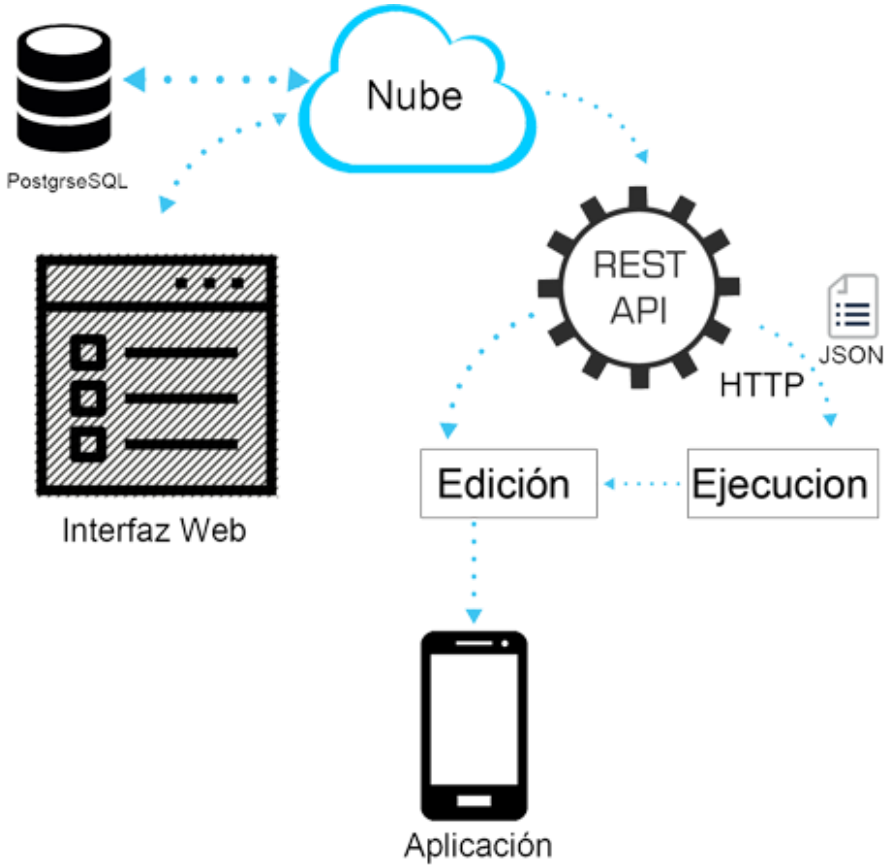


Figura 3. Estructura de la solución final

La interfaz web de la plataforma dividirá por categorías los *assetbundles*. De cada uno de ellos se mostrará su información, permitiendo modificarla y eliminarla solo en caso de que el recurso haya sido cargado por el propio usuario. Solo el administrador de la plataforma va a tener la capacidad de crear, mostrar, modificar y eliminar (CRUD) las categorías. Además, la plataforma brinda una guía y las herramientas necesarias para la construcción y consumo de los recursos.

La plataforma de edición es la encargada de generar las aplicaciones empleando la información que brinda la interfaz web y la plataforma de ejecución. Las aplicaciones guardan en cache los recursos consumidos para evitar perdida de información en caso de fallo de la disponibilidad.

La plataforma de ejecución tiene como objetivo explorar los recursos que brinda la plataforma web sin necesidad de acceder a ella. Para ello la API REST utiliza el protocolo HTTP para intercambiar datos con el motor de juegos Unity3D. Así mismo las respuestas a las peticiones van a estar en lenguaje de intercambio de información, empleando en este caso JSON.

3.2.2 Descripción de la implementación

Basado en la estructura de solución descrita en el epígrafe anterior. Se implementaron 3 prototipos de solución que muestran las funcionalidades principales de las HU recolectadas.

El primero de los prototipos implementados fue el sistema web. Este sistema fue desarrollado con el *framework* Yii en su versión 2. Se implementaron las funcionalidades relacionadas con la organización y estructuración de los contenidos en la nube, así como la gestión de usuarios y sus contenidos correspondientes que se describen en las HU en el epígrafe 2.3.1. Al mismo tiempo se implementó el servicio web que ofrece los contenidos utilizando la estructura que se muestra en las figuras 4 y 5.

```
{"categories":
   [{"id":"integer", "name":"string", "photo":"filename"},
    {…}
   ]}
```

Figura 4. Estructura del JSON que expone la información de las categorías de *assetbundles*.

En la figura 4, se muestra la estructura del JSON que describe como se debe consumir la información relacionada con las categorías de los *assetbundles* que se encuentran en la nube. Mediante esta estructura se muestra una lista de las categorías de los *assetbundles* que existen en la nube, las cuales tiene un identificador numérico, un nombre y una dirección *URL* donde se encuentra una imagen que describe la categoría. Cuando se consume este servicio, es necesario consultar además el listado de *assetbundles* relacionados con una categoría. Esta información se consume en otro servicio que se describe en la figura 5.

En la figura 5 se describe mediante un JSON la estructura de información que tiene asociado cada *assetbundle* en la nube. Cada elemento tiene, el identificador de la categoría a la que pertenece, un nombre, descripción, una dirección *URL* de la imagen que lo describe visualmente, la *URL* donde se encuentra ubicado el archivo del *assetbundle* en la nube, el identificador del usuario que lo subió a la nube, la fecha de subida a la nube, si es público o no y el nombre del *asset*. La información que se muestra en las figuras 4 y 5, es la que será consumida por las plataformas de edición y ejecución respectivamente.

```
{"element":
 [{"id":"integer", "id_categories":"integer",
 "name":"string", "description":"string",
 "photo":"string(filename)", "file":"string(filename)",
 "id_usuario":"integer", "date_upload":"date",
 "public":"bool", "assetbundle":"string"},
  {...}
 ]}
```

Figura 5. Estructura del JSON que expone la información de los *assetbundles*.

El segundo prototipo implementado fue la plataforma de edición de los contenidos virtuales para Realidad Aumentada. La plataforma de edición es una solución o proyecto de Unity3D en la que se construye una escena de Realidad Aumentada basada en ARToolkit. En este proyecto se definen los marcadores de ARToolKit que se van a utilizar en la aplicación final y para asociar los contenidos virtuales que se van a mostrar, se define el script "*CachingLoadExample*". Este script define la *URL* y el nombre del *asset* que debe mostrar y es el encargado de consumir el servicio web para descargar el *assetbundle* correspondiente y visualizarlo en el contexto de Realidad Aumentada. La información de la *URL* del *asset* que se debe mostrar puede ser consultada desde el sistema web o con la plataforma de ejecución. Ver figura 6.

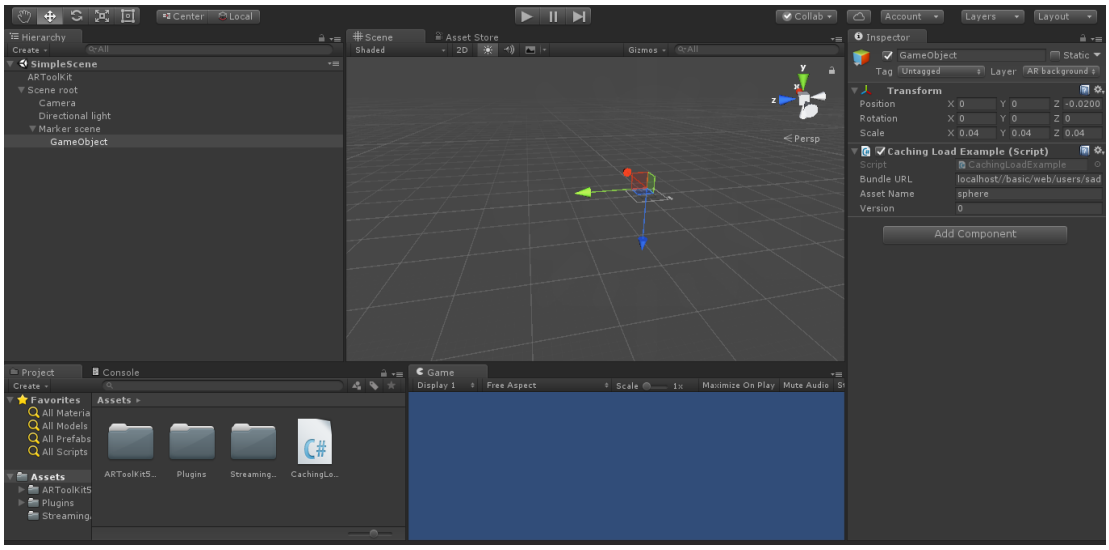


Figura 6. Plataforma de edición.

El tercer prototipo implementado fue la plataforma de ejecución la cual, sigue una lógica similar al sistema web. Esta plataforma no incluye la parte de la gestión de los contenidos virtuales, ni la gestión de usuario. Este se constituye como una herramienta de consulta y visualización de los *assetbundles* públicos que están en la nube y aprovecha las funcionalidades del motor de juegos Unity3D para proveer un entorno de edición que permita ajustar como se van a ver los elementos virtuales en la escena. La información que se obtiene luego del proceso de edición es la que se utiliza finalmente en la plataforma de edición para mostrar los contenidos de Realidad Aumentada de forma coherente. Ver figura 7.

Figura 7. Plataforma de ejecución.

3.3 Pruebas del Sistema

Luego de concluida la fase de implementación, se realizaron las pruebas de sistemas, las cuales posibilitan tanto a clientes como programadores encontrar errores para así poder comprobar si el producto final tiene la calidad requerida por el cliente. Existen varios tipos de pruebas las cuales se aplicaron a los prototipos implementados.

3.3.1 Pruebas de Aceptación

Las pruebas de aceptación (también conocidas como pruebas de aceptación del usuario) son un tipo de ensayos que se realzan con el fin de verificar si el producto ha sido desarrollado de acuerdo con las normas y criterios establecidos, y cumple con todos los requisitos especificados por el cliente. Este tipo de pruebas se llevan a cabo generalmente por un usuario o cliente.

Las pruebas de aceptación se consideran pruebas de caja negra, donde el usuario no evalúa el trabajo interno del sistema, sino evalúa el funcionamiento global del sistema y Verifica los resultados con los requisitos preestablecidos por ellos. Las pruebas de aceptación del usuario son consideradas una de las más importantes por el usuario antes de que el sistema es finalmente entregado (34).

Para la realización de las pruebas de aceptación se tuvieron en cuenta los siguientes aspectos:

- **Código**: Identifica la prueba en cuestión, incluye el número de HU y de la prueba.
- **HU**: Número de la HU a la cual pertenece.
- **Nombre**: Nombre del caso de prueba.
- **Descripción**: Acciones que debe realizar el sistema.
- **Condiciones de ejecución**: Describe las características y elementos en general que debe tener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución**: Incluye las entradas necesarias del sistema y los pasos necesarios para realizar el caso de prueba.
- **Resultados Esperados**: Respuesta que debe dar el sistema luego de aplicar el caso de prueba.

- **Resultado Obtenido**: Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba**: Clasificación de la prueba en satisfactoria o insatisfactoria.

A continuación, se muestran algunas de las pruebas de aceptación realizadas:

Pruebas de Aceptación	
Código**:** HU1_P1	HU: 1
Nombre: Crear Usuario	
Descripción: El usuario se podrá registrar en el sistema para acceder a la plataforma web.	
Condiciones de Ejecución: El usuario para autenticarse debe estar registrado en el sistema.	
Entrada: Datos personales del usuario necesarios para registrarse. Pasos de Ejecución: 1. Ir a la opción Registrarse. 2. Introducir los datos que se solicitan. 3. Hacer clic en el botón Registrase.	
Resultado Esperado: La aplicación muestra un mensaje al crear la cuenta con el texto “Usuario Registrado Satisfactoriamente”.	
Resultado Obtenido: La aplicación mostró un mensaje al crear la cuenta con el texto “Usuario Registrado Satisfactoriamente”.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 20. Prueba de aceptación para la HU “Crear Usuario”.

Pruebas de Aceptación	
Código: HU5_P1	HU: 5
Nombre: Crear A*ssetbundles*	
Descripción: El sistema debe permitir alojar los *assetbundles* en la nube.	
Condiciones de Ejecución: • El usuario tiene que estar autenticado.	
Entrada: Datos necesarios para crear un *assetbundle.* Pasos de Ejecución: 1. Ir a la opción *Assetbundles*. 2. Introducir los datos que se solicitan. 3. Hacer clic en el botón Crear.	
Resultado Esperado: La aplicación muestra un mensaje al cargar el *assetbundle* con el texto “Assetbundle Cargado Satisfactoriamente”.	
Resultado Obtenido: La aplicación no almacena los *assetbundles* en la nube.	
Evaluación de la Prueba: No Satisfactoria.	

Tabla 21. Prueba de aceptación para la HU “Crear *assetbundles*”.

Pruebas de Aceptación	
Código: HU9_P1	HU: 9
Nombre: Consultar *assetbundles*	

Descripción: El sistema debe ofrecer tanto la *URL* del *assetbundle* como una opción de descarga. Además de un servicio web para visualizar los *assetbundles* alojados en la nube.
Condiciones de Ejecución: • El usuario tiene que estar autenticado. • El usuario solo podrá consultar los *assetbundles* públicos y los privados a los que es añadido.
Entrada/Pasos de Ejecución: 1. Ir a la opción Categorías. 2. Seleccionar la categoría deseada. 3. Seleccionar el *assetbundle* deseado. 4. Realizar la acción de copiar el *URL* del *assetbundle* mediante el botón Copiar *URL* (*clipboard*). Si el usuario lo desea puede descargar el *assetbundle*, dando clic en el botón Descargar. 5. Para visualizar los *assetbundles* abrir el servicio web en Unity3D. 6. Hacer clic en el botón Play de Unity3D. 7. Seleccionar la categoría deseada. 8. Hacer clic en el botón *assetbundle* del recursos deseado.
Resultado Esperado: Los recursos se puedan visualizar en Unity3D, la aplicación brinde la *URL* y descargue los *assetbundles* correctamente.
Resultado Obtenido: Los recursos se visualizan correctamente en Unity3D, la aplicación brindó la *URL* y descargó los *assetbundles* correctamente.
Evaluación de la Prueba: Satisfactoria.

Tabla 22. Prueba de aceptación para la HU “Consultar *assetbundles*”.

Pruebas de Aceptación	
Código: HU10_P1	**HU:** 10
Nombre: Crear Categorías.	
Descripción: El sistema debe permitir crear las categorías por las que se dividen los *assetbundles*.	
Condiciones de Ejecución: • El usuario tiene que estar autenticado. • El usuario tiene que ser administrador del sistema.	
Entrada: Datos necesarios para crear una categoría. Pasos de Ejecución: 1. Ir a la opción Gestionar Categorías. 2. Hacer clic en el botón Crear Categorías 3. Introducir los datos solicitados. 4. Hacer clic en el botón Crear.	
Resultado Esperado: La aplicación muestra un mensaje al crear una categoría con el texto “Categoría Creada Satisfactoriamente”.	
Resultado Obtenido: La aplicación mostró un mensaje al crear una categoría con el texto “Categoría Creada Satisfactoriamente”.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 23. Prueba de aceptación para la HU “Crear Categorías”.

Se utilizaron, para aplicar los casos de pruebas, dos iteraciones para evaluar todas las HU, dado que en la primera iteración hubo evaluaciones de prueba insatisfactorias, fue necesario una segunda iteración para solucionar los errores. Ver Gráfico 1.

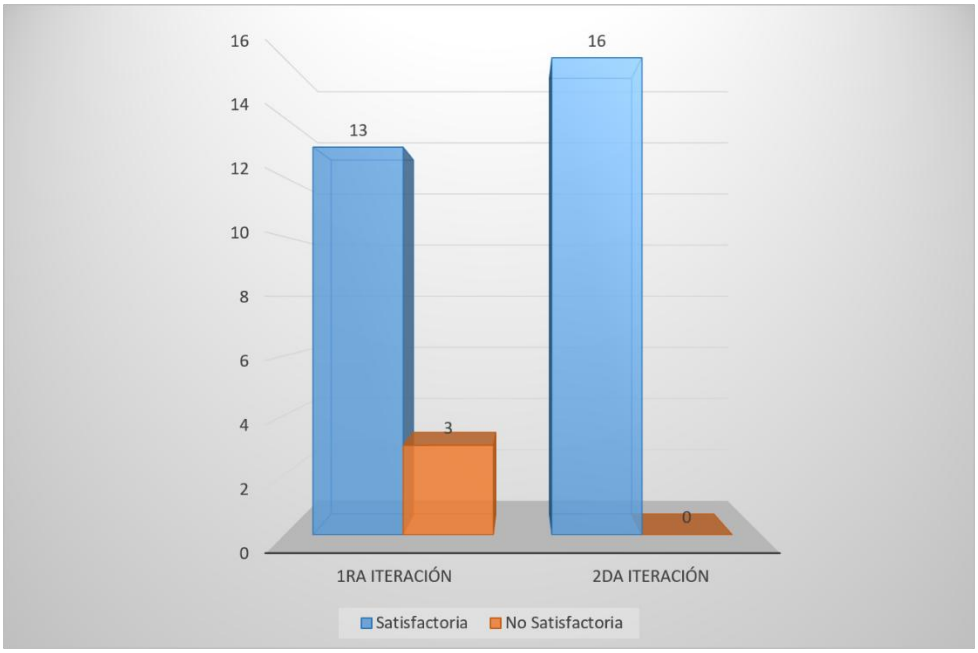


Gráfico 1. Estadísticas de las pruebas de aceptación.

3.3.2 Pruebas Unitarias

Las pruebas unitarias buscan aislar cada parte del programa y mostrar que las partes individuales son correctas, proporcionando cinco ventajas básicas:

Fomentar el cambio, las pruebas unitarias facilitan la reestructuración del código (refactorización), puesto que permiten hacer pruebas sobre los cambios y verificar que las modificaciones no han introducido errores (regresión).

Simplificar la integración, permiten llegar a la fase de integración asegurando que las partes individuales funcionan correctamente. De esta manera se facilitan las pruebas de integración.

Separar la interfaz y la implementación, la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro.

Menos errores y más fáciles de localizar, las pruebas unitarias reducen la cantidad de errores y el tiempo en localizarlos.

Mejorar el diseño, la utilización de prácticas de diseño y desarrollo dirigida por las pruebas permite definir el comportamiento esperado en un paso previo a la codificación.

Puede ser la forma más simple de verificar el funcionamiento, en situaciones como el desarrollo de una API o un componente que brinda servicios del cual no se cuenta aún con un cliente para consumirlos (35).

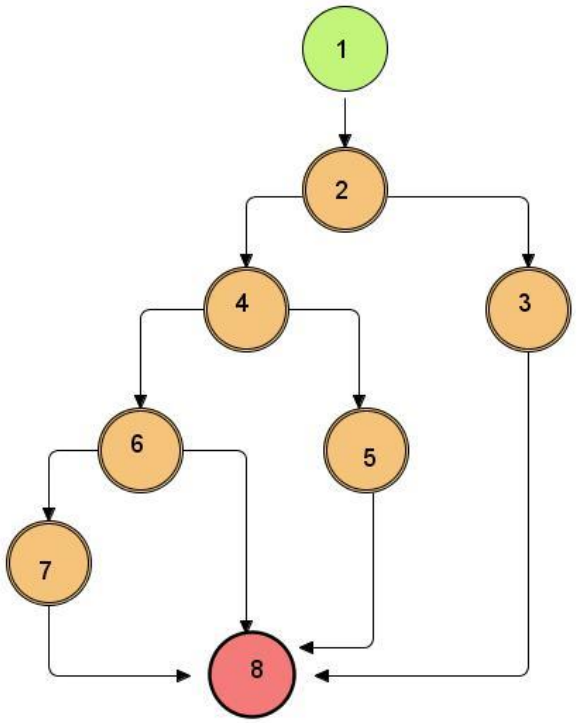
Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que utiliza la estructura de control del diseño para derivar los casos de pruebas.

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecutan todos los bucles en sus límites.
- Se utilizan todas la estructuras de datos internas (36).

Método: actionCreate()	Grafo Resultante

Enumerator DownloadAndCache (){
(1) yield return www;
(2) if (www.error! = null)
(3) throw new Exception ("WWW download had an error:" + www.error);
(4) AssetBundle bundle = www.assetBundle;
(4) if (AssetName == "")
(5) Instantiate(bundle.mainAsset);
else {
(6) if (obj == null)
{
(7)obj = Instantiate (bundle.LoadAsset (AssetName)) as GameObject;
	

(7)obj.transform.SetParent (transform); **(7)**obj.transform.localScale = new Vector3(0.5f,0.5f,0.5f); **(7)**obj.layer = 8; **(7)**obj.transform.position = new Vector3(0, 0, 0); } (8)}	
Complejidad Ciclomática:	Caminos:
V (G)= # de regiones = 4 V (G)=A-N+2 = 10-8+2 = 4 V (G)= P+1= 3+1 = 4	1. {1, 2, 3, 4} 2. {1, 2, 4, 5, 8} 3. {1, 2, 4, 6, 8} 4. {1, 2, 4, 6, 7, 8}

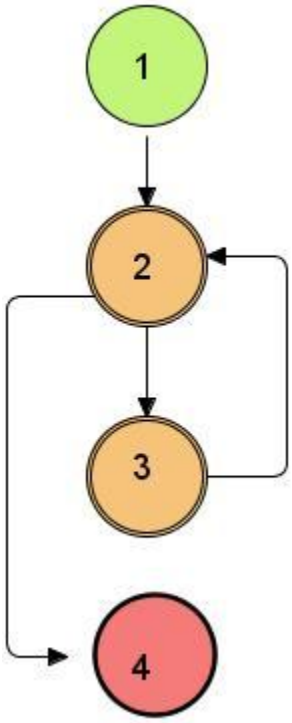
Tabla 24. Prueba de Caja Blanca al método DownloadAndCache ().

Número de Camino	Entradas	Objetivo	Resultado Esperado
1	error= ”error al cargar” *AssetName*=” Ejemplo” obj=*null*	Probar que devuelve el sistema al ocurrir un error	throw new Exception="WWW download had an error:" + www.error) Total. entradas=1 Total. validos=1
2	error=”” AssetName=””	Probar que devuelve el sistema si	ArgumentException: The thing you want to instantiate is null.

	obj=*null*	A*ssetName* está vacío	Total. entradas=1 Total. validos=1
3	error=”” A*ssetName*=” Ejemplo” obj=isA*ctive*	Probar que ocurre si obj está activado.	No instancia el recurso en pantalla Total. entradas=1 Total. validos=1
4	error=”” A*ssetName*=” Ejemplo” obj=isA*ctive*	Probar que ocurre si obj es nulo.	Se instancia el recurso en pantalla Total. entradas=1 Total. validos=1

Tabla 25. Caso de pruebas a los caminos de la clase DownloadAndCache

Método: Processjson ()	Grafo Resultante

private void Processjson(string jsonString) {
(1) JsonData jsonvale = JsonMapper.ToObject(jsonString);
(1)elementName = new ArrayList ();
(1)urlFile = new ArrayList ();
(1) MyTextureURL = new ArrayList ();
(1)idCategorie = new ArrayList ();
(2)for(int i = 0; i<jsonvale["element"].Count; i++)
{
(3)elementName.Add (jsonvale["element"][i]["name"].ToString());
(3)urlFile.Add(jsonvale["element"][i]["file"].ToString());
(3) MyTextureURL.Add(jsonvale["element"][i]["photo"].ToString());
(3)idCategorie.Add(jsonvale["element"][i]["id_categories"].ToString()
	

} (4)}	
Complejidad Ciclomática:	Caminos:
V (G)= # de regiones = 2 V (G)=A-N+2 = 6-6+2 = 2 V (G)= P+1= 1+1 = 2	1. {1,2,4,5,6} 2. {1,2,3,2,4,5,6}

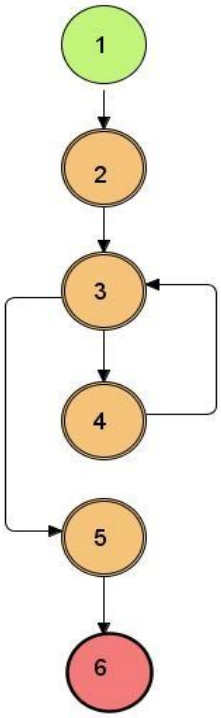
Tabla 26. Prueba de Caja Blanca al método processjson.

Número de Camino	Entradas	Objetivo	Resultado Esperado
1	Tabla *element* vacía	Probar como quedan los arreglos pasando una vez por el bucle.	Tamaño. Arreglos=0 Total. entradas=1 Total. validos=1
2	Tabla *element* con los datos de 50 *assetbundles*	Probar como quedan los arreglos pasando 50 veces por el bucle.	Tamaño. Arreglos=50 Total. entradas=50 Total. validos=50

Tabla 27. Caso de pruebas a los caminos de la clase processJson

Método: OnGUI()	Grafo Resultante

public void OnGUI(){
(1)if (readjsonElement._bWindowActive == false && categoriesName != null) {
(2)int sumY = 20;
(3)for (int i = 0; i < categoriesName.Count; i++) {
(4)sumY = sumY + 20;	

(4)if (GUI.Button (new Rect (5, sumY, 100, 20),categoriesName[i].ToString ())) {
(5)id = idCategories [i].ToString ();
(5)entryCategories = true;
}
}
(6)}
	

Complejidad Ciclomática:	Caminos:
V (G)= # de regiones = 2 V (G)=A-N+2 = 6-6+2 = 2 V (G)= P+1= 1+1 = 2	1. {1,2,3,5,6} 2. {1,2,3,4,3,5,6}

Tabla 28. Prueba de Caja Blanca al método OnGUI.

Número de Camino	Entradas	Objetivo	Resultado Esperado
1	categoriesName.Count=0	Probar que ocurre si *categoriesName* está vacío.	Los botones de las categorías no se muestran en pantalla Total. entradas=1 Total. validos=1

2	categoriesName.Count=10	Probar que ocurre si *categoriesName* tiene valor de 10.	Los botones de las categorías se muestran en pantalla Total. entradas=10 Total. validos=10

Tabla 29. Caso de pruebas a los caminos de la clase OnGUI

3.3.3 Pruebas de Rendimiento

El servicio de pruebas de rendimiento de software se centra en determinar la velocidad con la que el sistema, realiza una tarea en las condiciones particulares del escenario de pruebas. Este servicio ayuda a detectar los cuellos de botellas de la aplicación, previendo que los usuarios no sufran un mal funcionamiento (37).

Para las pruebas de rendimiento se empleó la herramienta Apache Jmeter, la cual es una herramienta de carga para llevar acabo simulaciones sobre cualquier recurso de software.

Inicialmente diseñada para pruebas de estrés en aplicaciones web, hoy día, su arquitectura ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en bases de datos, programas en perl, requisiciones ftp. Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de requisiciones que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción.

En este sentido, simula todas las funcionalidades de un Navegador (*Browser*), o de cualquier otro cliente, siendo capaz de manipular resultados en determinada requisición y reutilizarlos para ser empleados en una nueva secuencia (38).

El sistema web fue sometido a una prueba de estrés, fue necesario realizar 3 iteraciones para lograr un mejor rendimiento del tiempo de respuesta (milisegundos) del sistema. En cada iteración se empleó 3 grupos de hilos diferentes, contando con 50, 150 y 500 usuarios respectivamente, realizado peticiones HTTP (**Gráfico 2**). El servidor que hospedaba el sistema durante las pruebas de rendimiento presenta las siguientes propiedades:

- Tarjeta Madre: h81-m1
- Microprocesador: Core i5 4440

- RAM: 4 GB
- HHD: 1 TB

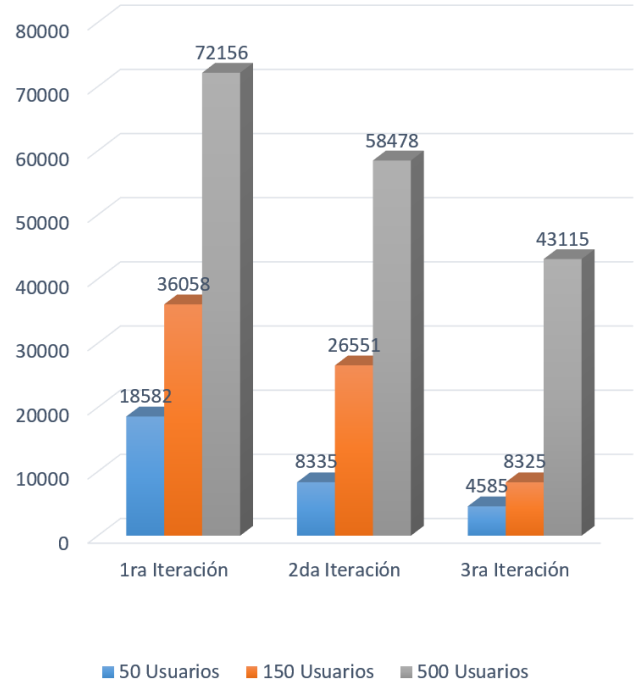


Gráfico 2. Estadísticas de las pruebas de rendimiento.

3.4 Análisis de resultados

Para la solución del problema planteado anteriormente, se desarrolló una plataforma web para la gestión de los contenidos de las aplicaciones de Realidad Aumentada. Una plataforma de ejecución donde los usuarios podrán visualizar los recursos alojados en la nube. Y una plataforma de edición para la construcción de las aplicaciones de Realidad Aumentada, empleando la *URL* obtenida en la plataforma web o en la plataforma de ejecución.

Para comprobar si la solución final cumple con el objetivo de reducir el espacio en disco de las aplicaciones de Realidad Aumentada, se generó dos aplicaciones de prueba para dispositivos Androides en la plataforma de edición. En la primera, los recursos necesarios para el funcionamiento de la aplicación van incluidos dentro de su archivo de instalación. En la segunda, los recursos son distribuidos a través del servicio en la nube. También se le

realizaron este tipo de pruebas a videojuegos con Realidad Aumentada creados en el centro Vertex. El resultado de la prueba se puede observar en la tabla 30.

Nombre	Tamaño en disco de la Primera Aplicación	Tamaño en disco de la Segunda Aplicación	Tamaño en disco de los recursos
Aplicación de Prueba	21.1 MB	16.3 MB	4.8 MB
Aventuras en la Manigua	32.6 MB	30.2 MB	2.4 MB
Especies Invasoras	95.6 MB	90.1 MB	5.5 MB

Tabla 30. Comparación entre una aplicación que incluye sus recursos en su archivo de intalación y una aplicación que sus recursos son distribuidos a través del servicio en la nube.

Como se observa en la tabla 30, las aplicaciones que sus recursos son distribuidos a través del servicio en la nube ocupan menos espacio en disco que las aplicaciones que incluyen sus recursos en su archivo de instalación.

Como valor agregado a la solución final, a la plataforma web se le pueden agregar funcionalidades para que funcione como un *Asset Store*. Un *Asset Store,* por ejemplo, el de Unity3D, es el hogar de una creciente biblioteca de *assets* comerciales y gratuitos creados por Unity Technologies y miembros de la comunidad. Estos *assets* son accesibles desde una interfaz simple dentro del Editor Unity3D (39). Para que la plataforma web funcione como un *Asset Store*, es necesario validar que los usuarios puedan almacenar en la nube *assets* y agregar categorías para las texturas, animaciones, proyectos completos, tutoriales y extensiones del editor. Además de permitir que los *assets* sean descargados e importados directamente en los proyectos.

3.5 Conclusiones Parciales

Luego de analizados los temas abordados durante el presente capítulo, se arribó a las siguientes conclusiones:

- El tamaño que ocupan en disco las aplicaciones de Realidad Aumentada disminuye, si los recursos necesarios para su correcto funcionamiento son distribuidos a través de un servicio en la nube.
- Las pruebas de rendimiento arrojaron que el servidor donde están alojados los recursos, no cuenta con las propiedades necesarias para el acceso de 500 o más usuarios.

Conclusiones Generales

Con la presente investigación se dio cumplimiento al objetivo propuesto y se arribó a las siguientes conclusiones:

- Se implementó una solución basada en la nube que consume los recursos virtuales a través de servicios web, lo cual permitió desagregar los contenidos virtuales del instalador de la aplicación, con la consiguiente reducción de tamaño del archivo de instalación de la aplicación.

- Se implementaron dos variantes de aplicación para consumir el servicio web, una en tiempo de ejecución y otra en tiempo de edición. La primera funciona como una búsqueda de catálogos de modelos en la nube, que permite inspeccionar modelos virtuales en el contexto de la Realidad Aumentada. La segunda permite generar aplicaciones personalizadas y a su vez puede ser utilizado como un *Asset Store*.

Recomendaciones

A continuación, se enuncian algunas recomendaciones orientadas a trabajos futuros:

- Comenzar la generación de un catálogo con las escenas y recursos producidos por el centro Vertex de forma que exista disponibilidad para los que desarrollen aplicaciones que incluyan esta solución.
- Desarrollar una aplicación de reproducción de contenidos previamente configurados que utilice esta solución para mostrar los productos desarrollados por el centro.
- Agregar funcionalidades para que la plataforma funcione también como un *Asset Store*.

Referencias Bibliográficas

1. iiemd. que-es-dropbox. *iiemd.* [En línea] 2015. [Citado el: 11 de Junio de 2017.] https://iiemd.com/dropbox/que-es-dropbox.

2. ticportal. amazon-web-services. *ticportal.* [En línea] 2014. [Citado el: 11 de Junio de 2017.] https://www.ticportal.es/temas/cloud-computing/amazon-web-services.

3. emprendices. que-es-google-cloud-platform. *emprendices.* [En línea] 29 de Octubre de 2014. [Citado el: 11 de Junio de 2017.] https://www.emprendices.co/que-es-google-cloud-platform/.

4. azure. what-is-azure. *azure.* [En línea] 2017. [Citado el: 11 de Junio de 2017.] https://azure.microsoft.com/es-es/overview/what-is-azure/.

5. computacion-nube-ejemplos. *hosting-alojamiento.* [En línea] [Citado el: 20 de Abril de 2017.] https://www.hosting-alojamiento.com/cloud/computacion-nube-ejemplos/.

6. Presmanes, Jorge Luis López. aplicacion-de-la-computacion-en-nube-en-la-gestion-de-la-biblioteca-virtual-de-la-ecured-ver-2-0. *biblat.* [En línea] 2013. [Citado el: 20 de Abril de 2017.] http://biblat.unam.mx/es/revista/ciencias-de-la-informacion/articulo/aplicacion-de-la-computacion-en-nube-en-la-gestion-de-la-biblioteca-virtual-de-la-ecured-ver-2-0.

7. Herrero, Manuel Ibáñez. arToolkit/Memoria%20ARToolkit.pdf. *disca.* [En línea] 2012. [Citado el: 20 de Noviembre de 2016.] http://www.disca.upv.es/magustim/val/pfcs_anteriors/arToolkit/Memoria%20ARToolkit.pdf.

8. Palazuelos, Félix. [En línea] 26 de Julio de 2016. [Citado el: 12 de Junio de 2017.] http://tecnologia.elpais.com/tecnologia/2016/07/12/actualidad/1468336791_763102.html.

9. Lapuente, María Jesús Lamarca. serv_internet. *hipertexto.* [En línea] [Citado el: 20 de Enero de 2017.] http://www.hipertexto.info/documentos/serv_internet.htm.

10. P.Mell, T.Grace. *The NIST Definition of Cloud Computing.* s.l. : NIST Special Publication 800-145, 2011.

11. Duarte, Eugenio. que-es-la-computacion-en-la-nube. *capacityacademy.* [En línea] 3 de Julio de 2012. [Citado el: 12 de Noviembre de 2016.] http://blog.capacityacademy.com/2012/07/03/que-es-la-computacion-en-la-nube/.

12. culturacion. *que-es-y-para-que-sirve-un-web-service.* [En línea] [Citado el: 15 de Noviembre de 2016.] http://culturacion.com/que-es-y-para-que-sirve-un-web-service/.

13. Chakray, Webmaster. cuales-son-las-ventajas-de-una-api-rest. *chakray.* [En línea] 9 de Febrero de 2017. [Citado el: 1 de Junio de 2017.] http://www.chakray.com/cuales-son-las-ventajas-de-una-api-rest/.

14. Fernando. Quien_esta_usando_ya_la_REST_API. *ayudawp.* [En línea] 25 de Enero de 2016. [Citado el: 1 de Junio de 2017.] https://ayudawp.com/wordpress-rest-api-que-es-como-funciona/#Quien_esta_usando_ya_la_REST_API.

15. NIELSEN, JAKOB. *Usabilidad, Diseño de Sitios Web.*

16. que-es-una-interfaz. *guiadigital.* [En línea] Unidad de Modernización y Gobierno Digital, Ministerio de Secretaría General de la Presidencia. [Citado el: 16 de Noviembre de 2016.] http://www.guiadigital.gob.cl/articulo/que-es-una-interfaz.

17. Mejía, Óscar Ávila. *Computación en la Nube.* s.l. : Depto. de Ingeniería Eléctrica, 2011.

18. Cruz, Andres. realidad-aumentada-con-vuforia. *desarrollolibre.* [En línea] 14 de Enero de 2014. [Citado el: 20 de Enero de 2017.] http://www.desarrollolibre.net/blog/tema/73/android/realidad-aumentada-con-vuforia.

19. Villamandos, Íñigo Sánchez. wikitude-una-potente-aplicacion-de-realidad-aumentada. *digiworks.* [En línea] 8 de Marzo de 2012. [Citado el: 20 de Enero de 2017.] http://www.digiworks.es/2012/03/08/wikitude-una-potente-aplicacion-de-realidad-aumentada/.

20. VisionStar Information Technology. Getting-Started-with-EasyAR. *easyar.* [En línea] 2016. [Citado el: 12 de abril de 2017.] https://www.easyar.com/view/docs/Getting-Started/Getting-Started-with-EasyAR.html.

21. Luttecke, Camilo. sabes-que-es-unity-descubrelo-aqui. *deideaaapp.* [En línea] [Citado el: 20 de Noviembre de 2016.] https://deideaaapp.org/sabes-que-es-unity-descubrelo-aqui/.

22. Unity. AssetBundlesIntro. *docs.unity3d.* [En línea] 2016. [Citado el: 16 de Noviembre de 2016.] https://docs.unity3d.com/es/current/Manual/AssetBundlesIntro.html.

23. Borja, Jorge. definicion-c. *programacioncsharp2014.* [En línea] 2 de Diciembre de 2013. [Citado el: 13 de Enero de 2017.] http://programacioncsharp2014.blogspot.com/2013/12/definicion-c.html.

24. Tecnología, H. html. *conceptodefinicion.* [En línea] 26 de Enero de 2015. [Citado el: 13 de Enero de 2017.] http://conceptodefinicion.de/html/.

25. P, General. php. *conceptodefinicion.* [En línea] 16 de Noviembre de 2014. [Citado el: 13 de Enero de 2017.] http://conceptodefinicion.de/php/.

26. Commons, Creative. Iagp2. *um.* [En línea] 30 de Diciembre de 2006. [Citado el: 10 de Enero de 2017.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

27. Flores, Lic. Ervin. XP---Extreme-Programing. *ingenieriadesoftware.* [En línea] [Citado el: 11 de Enero de 2017.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

28. Martinez, Rafael. postgresql. *sobre_postgresql.* [En línea] 2 de octubre de 2010. [Citado el: 18 de febrero de 2017.] http://www.postgresql.org.es/sobre_postgresql.

29. yii2-framework. *intro-yii.* [En línea] [Citado el: 18 de febrero de 2017.] http://yii2-framework.readthedocs.io/en/stable/guide-es/intro-yii/.

30. Ledesma, Anibal. prezi. *wampserver.* [En línea] 29 de Noviembre de 2013. [Citado el: 19 de Febrero de 2017.] https://prezi.com/h0j-_95vhkj2/que-es-wampserver/.

31. Pater, Lis. slideshare. *metodologias-agiles-xp.* [En línea] 6 de Marzo de 2013. [Citado el: 28 de Febrero de 2017.] https://www.slideshare.net/LisPater1/metodologias-agiles-xp.

32. Zambrano, Omar. scribd. *Que-son-y-para-que-sirven-las-tarjetas-CRCs-docx.* [En línea] [Citado el: 1 de Marzo de 2017.] https://es.scribd.com/document/207429482/Que-son-y-para-que-sirven-las-tarjetas-CRCs-docx.

33. Alvarez, Miguel Angel. desarrolloweb. *que-es-mvc.* [En línea] 2 de Enero de 2014. [Citado el: 1 de Marzo de 2017.] https://desarrolloweb.com/articulos/que-es-mvc.html.

34. *globetesting.* [En línea] 29 de Agosto de 2012. [Citado el: 15 de Abril de 2017.] https://www.globetesting.com/pruebas-de-rendimiento/.

35. Sommariva, Andrés. 74-unit-test-part1-mock. *microgestion.* [En línea] [Citado el: 7 de junio de 2017.] http://www.microgestion.com.ar/index.php/mg-developers/articulos/74-unit-test-part1-mock.

36. Carmona, Cristal Ramirez. prueba-caja-blanca. *slideshare.* [En línea] 2 de Julio de 2013. [Citado el: 15 de Abril de 2017.] https://es.slideshare.net/cristalramirezcarmona/prueba-caja-blanca.

37. Glove. pruebas-de-rendimiento. *globetesting.* [En línea] [Citado el: 16 de Abril de 2017.] https://www.globetesting.com/pruebas-de-rendimiento/.

38. basico. *osmosislatina.* [En línea] [Citado el: 15 de Abril de 2017.] https://www.osmosislatina.com/jmeter/basico.htm.

39. Unity. AssetStore. *docs.unity3d.* [En línea] 2017. [Citado el: 11 de Junio de 2017.] https://docs.unity3d.com/es/current/Manual/AssetStore.html.