



Universidad de Valladolid

E.T.S Ingeniería Informática

Trabajo Fin De Grado

Grado en Ingeniería Informática

**Gestor de contenidos de modelado 3D y una
aplicación Android de realidad aumentada para su
visualización.**

Autor:

D. Iván Martín Pérez



Universidad de Valladolid

E.T.S Ingeniería Informática

Trabajo Fin De Grado

Grado en Ingeniería Informática

**Gestor de contenidos de modelado 3D y una
aplicación Android de realidad aumentada para su
visualización.**

Autor:

D. Iván Martín Pérez

Tutor:

D. Valentín Cardenoso Payo

Resumen

Este proyecto ha sido realizado por el autor como Trabajo de Fin de Grado del Grado en Ingeniería Informática de la Universidad de Valladolid.

Su objetivo es el desarrollo y puesta en marcha de una aplicación web para el almacenaje de modelos 3D, que facilite el almacenamiento, clasificación y distribución para su posterior visualización en dispositivos Smartphone. Por consiguientes, se ha desarrollado tanto una aplicación web como una aplicación Android para móvil que permite la visualización de modelos 3D, tanto estáticos como soportando Realidad Aumentada.

En la aplicación web se gestiona el contenido de modelado, teniendo en cuenta las necesidades de los clientes. Por otro lado, la aplicación móvil dispone de dos visualizadores de modelo 3D, uno con realidad aumentada y otro de visualizador 3D de modelos. Las aplicaciones se comunican a través de internet para que así el cliente pueda interactuar entre ellas.

La meta de este proyecto es realizar un producto útil para el cliente y utilizar la tecnología actual para su implementación. La aplicación web se instalarán en un servidor y la aplicación móvil se subirá a la plataforma de distribución digital de Google Play Store.

Agradecimientos

Me gustaría dar mi agradecimiento a varias personas, gracias a las cuales este Trabajo de Fin de Grado se ha convertido en una realidad.

En primer lugar , me gustaría agradecer a todos los docentes que me han impartido clase a lo largo del Grado, por aportarme conocimientos, formación y dedicación, lo cual ha permitido mi desarrollo laboral y personal. También me gustaría resaltar mi gratitud a Mario Corrales Astorgano, por su ayuda y colaboración dentro de la parte que hace referencia mi proyecto a la realidad aumentada. Tampoco me puedo olvidar de mi tutor, Valentín Cardeñoso Payo, porque gracias a su esfuerzo y a su acción como tutor he podido llevar a cabo este proyecto.

Por otro lado, dentro del ámbito personal, tengo que agradecer el apoyo que me han ofrecido, principalmente a mis padres, los cuales siempre han estado ahí para acompañarme a lo largo de todos mis años de universidad, en los buenos y en los malos momentos, apoyándome de forma incondicional. También me gustaría mencionar a Sandra Becerril, por ayudarme a crecer como persona, animarme y confiar en mí.

Y por último dar las gracias a Eduardo Delgado y Jesús Amor, por confiar en mí y dejarme demostrar día a día mis cualidades como informático trabajando para hacer posible su proyecto.

Prólogo

En la actualidad, el Smartphone es un dispositivo indispensable para las personas y ya no está concebido solo para realizar llamadas sino que también puede hacer fotos, estar en las redes sociales, reproducir música etc. Cada año se innova en este campo buscando su mejora, además de nuevas funcionalidades en los dispositivos. También las prestaciones de estos se ven aumentadas de forma rápida y constante, ya sea en forma de software o hardware.

Las aplicaciones móviles están en auge desde hace unos años, y proporcionándonos servicios como navegador GPS (Google maps), agenda (Google calendar), entretenimiento (Clash of Clans) redes sociales (Facebook) v.g. Todos los días se realizan millones de descargas de estas aplicaciones y se llegan a mover importantes sumas de dinero en todo el mundo, donde las empresas buscan su beneficio. En España la cifra de descarga diaria asciende a 4 millones. (1)

La realidad aumentada se ha asentado en la sociedad, siendo usada para visualizar información, animaciones, videojuegos, o incluso para ayudar a personas discapacitadas. (2; 3) En la actualidad, la sociedad tiene un gran desconocimiento sobre este tema, han oído hablar de ello en algunos medios de comunicación, pero no han tenido un contacto directo con ello.

Estamos en la era del *Internet de las cosas*, donde cada dispositivo tiene algún tipo de sensor que capta información y es almacenada en algún servidor del mundo. Tal hecho, provoca que el almacenamiento y el análisis de datos se haya exponenciado durante la última década, con el objetivo de poder procesar esta información y hacer un análisis de ella.

A medida que el volumen de contenidos gráficos 3D, tanto estáticos como vinculados a aplicaciones de realidad aumentada, crece, aumenta el interés por disponer de servicios de almacenamiento, clasificación y descarga de estos contenidos, así como de apps que faciliten la explotación de los mismos. Este trabajo supone un paso en esta dirección y con él se pretenden alcanzar los objetivos y competencias asociados al Trabajo de Fin de Grado del Grado en Ingeniería Informática.

Indicé General

1	Introducción.....	5
1.1	Descripción	5
1.2	Necesidades	5
1.3	Objetivos	8
1.4	Metodología	8
1.5	Resumen del contenido.....	11
2	Planificación	13
2.1	Definición de tareas.....	13
2.2	Secuenciación de tareas	14
2.3	Asignación de tiempos	15
2.4	Gestión del riesgo	16
2.5	Recursos empleados para el proyecto	19
3	Análisis.....	20
3.1	Análisis de usuarios	20
3.2	Definición de los personajes.....	21
3.3	Análisis de tareas.....	22
3.4	Modelo de dominio	23
3.5	Requisitos funcionales	24
3.6	Requisitos no funcionales.....	25
3.7	Casos de uso	25
4	Diseño	38
4.1	Tecnología seleccionada.....	38
4.2	Arquitectura y patrones	41
4.3	Base de datos	51
4.4	Navegación en la aplicación.....	52
4.5	Diagramas de secuencias y diagramas de estado	56
5.	Implementación	57
5.1	Proceso de implementación	57
5.2	Implementación de casos específicos	57
5.3	Otras herramientas utilizadas.....	59
5.3.1	Foundation	59
5.3.2	JSC3D	59

5.3.3 PclZip	60
5.3.4 BreadCrumb	60
6 Pruebas de validación.....	61
6.1 Metodología.....	61
6.2 Validación de la aplicación.....	62
6.3 Resultados de las pruebas	75
6.4 Conclusiones.....	76
7 ARkivia viewer	77
7.1 Análisis	77
7.2 Diseño	82
7.3 Implementación	88
7.4 Pruebas de validación.....	89
8 Conclusión	92
8.1 Trabajo realizado	92
8.2 Trabajo futuro	93
9 Bibliografía.....	94
Anexo	97
A. Diagramas.....	97
B. Manuales de usuario	106
C. Manual de instalación	106
D. Contenido del CD-ROM	106

Indicé de figuras e ilustraciones

Figura 1 - Planificación de tareas secuencial del sistema ARkivia.....	14
Figura 2 - Planificación de tareas con varias personas trabajando en el sistema ARkivia.	14
Figura 3 - Planificación final de las tareas del sistema ARkivia.	15
Figura 4 - Modelo de dominio de ARkivia web.....	23
Figura 5 - Paquetes de casos de uso de ARkivia web.....	26
Figura 6 - Diagramas de casos de uso de gestión de usuarios.	26
Figura 7 - Diagramas de casos de uso de gestión de álbum.	27
Figura 8 - Diagramas de casos de uso de gestión de modelo.	27
Figura 9 - Diagramas de casos de uso de gestión de complemento.....	28
Figura 10 - Diagramas de casos de uso de gestión de petición.....	28
Figura 11 - Diagrama de la arquitectura de 3 capas. (31).....	42
Figura 12 - Representación de la clase modelo en ARkivia web.	43
Figura 13 - Representación de la clase album en ARkivia web.	43
Figura 14 - Representación de la clase compelemento en ARkivia web.....	44
Figura 15 - Representación de la clase petición en ARkivia web.....	44
Figura 16 - Representación de la clase usuario en ARkivia web.	45
Figura 17 - Representación de la clase usuario en ARkivia web.	45
Figura 18 - Estructura de la carpeta principal de Yii.	46
Figura 19 - Estructura de la carpeta “protected” de Yii.	47
Figura 20 - Estructura de la carpeta “models” de Yii.....	48
Figura 21 - Estructura de carpetas “models” de Yii. (2, Pág. 37)	49
Figura 22 - Estructura de la carpeta “controllers” de Yii.....	49
Figura 23 - Jerarquía de herencia de los controladores.....	50
Figura 24 - Estructura de la carpeta “views” de Yii.	51
Figura 25 - Workflow de la área de usuario no registrado.	52
Figura 26 - Workflow de la sección desde el panel de control.	53
Figura 27 - Workflow de la sección de álbum.	53
Figura 28 - Workflow de la sección de modelo.	54
Figura 29 - Workflow de la sección complemento.	54
Figura 30 - Workflow de la sección galería.....	54
Figura 31 - Workflow de la sección de ver perfil.	55
Figura 32 - Workflow de la barra de menú superior.....	55
Figura 33 - Diagrama de estado de una petición en ARkivia web.....	56
Figura 34 - Estructura del directorio privado de los usuarios, modelos, complementos y peticiones.	58

Figura 35 - Diagrama de casos de uso de la aplicación de ARkivia viewer.	79
Figura 36 - Representación de la arquitectura de Android. (29)	82
Figura 37 - Diagrama de clases de la aplicación móvil.	84
Figura 38 - Estructura general de la aplicación móvil.	84
Figura 39 - Estructura de la aplicación de ARkivia propia.	85
Figura 40 - Estructura de los recursos de la aplicación de ARkivia viewer.	86
Figura 41 - Workflow de la aplicación web.	87

Ilustración 1 - Diagrama de Gantt del proyecto.	97
Ilustración 2 - Diagrama de secuencia de ver perfil de usuario.	98
Ilustración 3 - Diagrama de secuencia de cambiar contraseña.	99
Ilustración 4 - Diagrama de secuencia de borrado de cuenta.	100
Ilustración 5 - Esquema de la base de datos de la aplicación web.....	101
Ilustración 6 - Diagrama de secuencia de descargar modelo.....	102
Ilustración 7 - Diagrama de secuencia de descargar modelo.....	103
Ilustración 8 - Diagrama de secuencia de petición a API para descargar modelo.	104
Ilustración 9 - Diagrama de secuencia de añadir modelo una vez creado el modelo.	105

1 Introducción

1.1 Descripción

El sistema que vamos a implementar se divide en dos partes, por un lado tenemos el gestor de contenidos que será una aplicación web y daremos la posibilidad de guardar los modelos 3D a los usuarios. La segunda parte es una aplicación móvil para Android para la visualización de estos modelos.

El principal objetivo de este proyecto, es la implementación de una solución que cubre las necesidades del usuario del sistema y la superación de trabajo de fin de carrera. Para ello he realizado las 2 aplicaciones que se complementan, teniendo la capacidad de funcionar independientemente la una de la otra. Utilizamos el nombre de "ARkivia" o "ARkivia web" para nombrar a la aplicación web y "ARkivia viewer" para la aplicación móvil. A continuación mostramos una descripción breve de las 2 aplicaciones.

- ARkivia: es la aplicación web encargada de gestionar los modelados 3D creados por los usuarios. Permite la visualización, la organización, y el intercambio de los modelos entre los usuarios registrados.
- Arkivia Viewer: es la aplicación Android capaz de conectarse con la aplicación web ARkivia y descargarse los modelos para su posterior visualización ya sea en 3D o con realidad aumentada.

La parte principal del proyecto es que damos facilidades a los usuarios para la organización y compartición de sus modelos en la plataforma web así como poder ver los modelos de los demás y poder descargarlos.

La seguridad es importante en las dos aplicaciones, así como la protección de la comunicación entre ellas. Las dos aplicaciones usan una única base de datos y los tiempos de transferencia de datos deben de ser razonables.

La implementación y puesta en marcha de las aplicaciones, están comprendidas como parte del proyecto. Se instala el proyecto web en un servidor y la aplicación Android se publica en la plataforma de Google Play Store. Se dará por finalizado el proyecto una vez complementado el desarrollo y efectuado la comprobación de rendimiento.

1.2 Necesidades

En este apartado hacemos un análisis inicial de las necesidades del cliente para poder hacer un producto enfocado a este. Si no se define correctamente este apartado, el producto o prototipo será solo una aplicación funcional pero que sin utilidad alguna ya que no soluciona ningún problema del cliente. Por ello, vamos a analizar el cliente y sus necesidades para diseñar el producto conforme el resultado obtenido.

Al crear varias aplicaciones debo encontrar las necesidades de cada una. Por esta razón, he dividido este apartado en tres grupos. El primero, va relacionado con la necesidad de los usuarios de almacenar sus modelos de forma segura. En el segundo, analizo la necesidad de organizar los modelos, y por último, estudio la necesidad de poder visualizar modelos en cualquier momento. Las páginas que hemos analizado están en la bibliografía. (4; 5; 6; 7; 8)

Almacenamiento de los modelos

Los dispositivos de almacenamiento actuales más comunes, como pueden ser CD-ROM, DVD, memoria USB, discos duros externos, memoria SD etc. Estos dispositivos nos ofrecen una cantidad de memoria suficiente para almacenar modelos 3D, sin que la cantidad de espacio pueda ser una preocupación para el cliente. En cambio este hardware lo tenemos que administrar nosotros, guardar, conectar, reparar y limpiar.

Uno de los aspectos que preocupa a nuestros usuarios a la hora de almacenar modelos es la seguridad. En los dispositivos de almacenamiento que he nombrado anteriormente, la seguridad depende de nosotros. No todo el mundo es un experto en seguridad, por lo que debe delegar esta función al producto.

Uno de los problemas que puede surgir es que un ordenador no esté disponible, por la razón que sea. Para poder seguir utilizando el material almacenado, debemos guardar todos los modelos y pasar la información a un dispositivo que esté disponible. Sabemos que existe la necesidad de que la información esté almacenada, y además disponible para su uso en cualquier momento. Por lo tanto, es necesario que nuestro cliente tenga sus modelos en cualquier momento.

Es importante recordar que en el mundo del modelado 3D, él o los archivos que lo componen son cruciales, si extraviásemos alguno de ellos perderíamos el modelo, así como el todo el trabajo efectuado durante su construcción. En la actualidad existen una gran variedad de aplicaciones web que nos permiten el *Cloud Storage*, como pueden ser Google Drive, Dropbox, OneDrive o iCloud.

Teniendo en cuenta lo anteriormente mencionado, en ARkivia los modelos están almacenados en un servidor donde se aloja la aplicación web.

Organización de los modelos

Actualmente, por norma general, el cliente guarda los modelos en sus dispositivos ordenados en carpetas, la organización depende del propio usuario, por lo que dependerá de él determinar cómo ubicar la información para así encontrarlos con el paso del tiempo. Además del posible problema de organización del propio cliente, éste también puede tener varios ordenadores, como por ejemplo, un ordenador de sobremesa y un portátil, esto puede conllevar a que tenga los modelos duplicados o sin sincronizar.

Por ahora, la organización de los modelos puede ser de mayor o menor grado, dependiendo siempre del usuario y del número de modelo que posee. Hay gente que puede organizarlo en una carpeta todos los modelos o bien almacenarlo en una jerarquía de carpetas en función del proyecto, fecha de creación u criterio. Como ya he dicho, todo depende de la persona.

Valorando lo desarrollado a lo largo de este punto, la solución que aportaré es una jerarquía con un único nivel y un sistema de búsqueda de modelo, por nombre y/o categoría, que permita encontrar rápidamente un modelo en el sistema.

Visualizar los modelos en cualquier lado

Los clientes que tienen modelos 3D necesitan visualizar o mostrar con bastante frecuencia, estos modelos que han construido, por esta razón es necesario que dispongan del modelo en cualquier momento. En algunos casos no tenemos la posibilidad de mostrar esos modelos, ya sea porque no tenemos un dispositivo capaz de reproducir el modelo, o bien, no tenemos el modelo con nosotros.

Con la intención de solventar esta necesidad, mi proyecto utiliza una aplicación móvil para poder dotar al cliente de un servicio que pueda tener sus modelos 3D, dónde y cuándo quiera para poder visualizarlos.

Competencia y conclusiones

Viendo la competencia he podido apreciar que la mayoría de las aplicaciones de almacenamiento de modelos no están actualizadas o incluso abandonadas. Estas cuentan con algunas deficiencias, como es en la organización de las páginas y los catálogos. Varias de ellas tienen una lista interminable de secciones y subsecciones que nos confunde a la hora de hacer una búsqueda específica.

Los registros en estas páginas suelen ser largas y por lo general, piden información innecesaria para lo que vamos realizar. También existen otras aplicaciones que no necesitan registro y ofrecen una gran cantidad de modelos para su visualización y descarga.

Una de las características más importante de estas páginas es la cantidad de formatos que soporta. Cuantos más formatos tenga, más opciones podemos aportar, por lo que tendremos más posibilidades de que los clientes potenciales se queden en nuestra aplicación. En el futuro sería interesante hacer un estudio de los formatos más empleados en la actualidad para poder satisfacer a más usuarios, y así poder llegar a más gente.

Para poder mostrar nuestros modelos 3D en cualquier parte, se necesita un sistema ligero que tenga una pantalla y que esté disponible las 24 horas. El Smartphone es el elemento perfecto para la visualización de estos, ya que cumple con las características. Los nuevos móviles son capaces de procesar modelos 3D con fluidez de tamaño medio-alto, y son muchas las personas que llevan uno de estos encima a diario, esto nos facilita el poder llegar a más gente. Las pantallas que incorporan los Smartphone son de suficiente calidad para la visualización de nuestros objetos, por eso he desarrollado un proyecto vistoso, que destaque sobre los demás, incorporando la realidad aumentada con el fin de que el usuario desarrolle una experiencia mucho más completa.

Existen aplicaciones móviles que nos permiten cubrir algunas de estas necesidades, pero nos exigen llevar nuestros modelos 3D siempre en el móvil estos archivos, dependiendo de su calidad y/o tamaño, pueden llegar a ser una carga molesta para su uso habitual en un smartphone.

Teniendo en cuenta las necesidades que cubre el proyecto, existe una aplicación que ofrece todas estas necesidades sobre la que me baso e inspiro. El proyecto se llama, Augment, está realizado por 13 personas y lleva en desarrollo desde el 2009. En mi opinión una de las debilidades que tiene, es su elevado coste. (9)

1.3 Objetivos

Los objetivos iniciales del proyecto, divididos por aplicaciones son:

ARkivia:

- Crear una aplicación web dinámica capaz de gestionar usuarios y contenidos de modelados 3D.
- Desarrollar una aplicación útil y sencilla, facilitando su uso a los diferentes perfiles de usuarios, teniendo en cuenta sus conocimientos de informática.
- Buscar la rápida inicialización en ARkivia a través de un formulario reducido de registro.
- Gestionar los modelos 3D que el usuario suba a la plataforma ARkivia.
- Ofrecer un intercambio de modelos 3D entre usuarios, de forma fácil.
- Dotar al cliente de un apartado personal donde gestionar sus modelos, ubicados en un álbum.
- Permitir al cliente que comparta o privatice los modelos que ha subido a la plataforma.

ARkivia viewer:

- Crear aplicación móvil que permita a nuestros usuarios de ARkivia conectar con nuestros modelos 3D subidos a la aplicación web, y posteriormente visualizar esos modelos mediante la realidad aumentada.
- Sincronizar los archivos previamente subidos, tras autenticarse con la cuenta creada en la aplicación web de ARkivia.
- Guardar los modelos en el smartphone del usuario para así poder utilizarlos de modo offline.
- Catalogar los modelos con la misma jerarquía que en ARkivia, a través de álbum.
- Eliminar los archivos del lugar almacenado.
- Disponer de un apartado de ayuda para los usuarios.
- Permitir la utilización a aquellos usuarios que se descarguen la aplicación y no tengan cuenta en ARkivia a través de un usuario predeterminado con funciones limitadas.

1.4 Metodología

Esta sección trata sobre los procesos utilizados para realizar las tareas necesarias para la realización del proyecto. El proceso principal que utilizo es el incremental, con algunas ideas del proceso unificado de software. Este proceso, el incremental y como los demás procesos, crea todos los modelos de análisis y de diseño para así poder realizar la construcción y la transición más fácilmente.

Cada incremento tiene un ciclo de tareas y su objetivo es la creación de un prototipo del producto. Este modelo lo llaman “modelo de evolución de prototipo”. Seguir este desarrollo me permite una alta realimentación con el cliente además de saber si el producto que se está diseñando es lo que espera, y si se adapta a sus necesidades.

A medida que se van finalizando los prototipos, pasando por cada iteración, se va mejorando y pareciéndose más al producto final. Una iteración tiene tareas de análisis, diseño, implementación y pruebas, basándose en los casos de uso que hemos definido en cada una de estas iteraciones.

El proceso unificado lo utilizo solo en parte ya que no sigo sus procedimientos para dirigir mi proyecto, ni genero todos los entregables necesarios en cada una de las cuatro fases que tiene, pero si que defino los casos de uso, y género algunos documentos que giran en torno a ellos, como pueden ser los modelos de dominio, los diagramas de secuencia o los diagrama de clases. (10) Otra de las características que utilizo de este proceso, es que se centra en la arquitectura del sistema para proporcionar facilidad en los cambios, reutilización y claridad. La arquitectura debe de ser capaz de incorporar todos y cada uno de los casos de uso previamente definidos.

Los documentos que se han creado son:

- Análisis:
 - Requisitos funcionales y no funcionales.
 - Diagrama de caso de uso.
 - Modelo de dominio.
- Diseño:
 - Diagramas de clases.
 - Diseño de la base de datos.
 - Diagramas de secuencias.
 - Diagramas de estado.

Se utilizara el lenguaje de modelado UML para la realización de estos documentos. Este lenguaje gráfico es para visualizar, especificar, construir y documentar un sistema. Es un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (11)

Las iteraciones que voy a realizar son 6:

- **Iteración 1: Análisis previo**

En la primera iteración recopilamos información sobre las tecnologías que voy a utilizar, sus herramientas de desarrollo y los entorno de programación propio. Después, planificamos las tareas que debo realizar para completar el proyecto. Para finalizar, hago un primer análisis de posibles clientes de mi aplicación y un estudio sobre ellos.

- **Iteración 2: Prototipo aplicación web**

La segunda iteración consiste en la creación de un prototipo de la aplicación web. Empezamos haciendo el análisis, después el diseño, se implementan las funcionalidades básicas y se le presenta al cliente para obtener una realimentación.

- **Iteración 3: Prototipo mejorado de la aplicación web**

Añadimos a mi proyecto los nuevos casos de uso impuestos en la iteración anterior gracias al cliente. Realizamos de nuevo el análisis y el diseño correspondiente a los nuevos casos de uso, como su implementación y sus pruebas. También debemos modificar los casos de uso que se han modificado respecto a la anterior iteración. Concluimos con un prototipo avanzado, con los requisitos mínimos que necesitan los cliente.

- **Iteración 4: Prototipo aplicación móvil**

La cuarta iteración consiste en la creación de un prototipo de la aplicación móvil. Empezamos haciendo el análisis de los usuarios y de la tecnología a utilizar, después el diseño, se implementan las funcionalidades básicas y se le presenta al cliente para obtener una realimentación.

- **Iteración 5: Prototipo mejorado de la aplicación móvil**

Como en el caso de la web, añado los nuevos casos de uso descubiertos en la iteración anterior, gracias al cliente. Realizamos el análisis y el diseño correspondiente a los nuevos casos de uso, como su implementación y acabamos haciendo las pruebas para ver si funciona todo correctamente. También debo modificar los casos de uso de la anterior iteración. Concluimos con un prototipo avanzado con los requisitos expuestos por el cliente.

- **Iteración 6: Pruebas del sistema e implantación**

En esta iteración realizamos las pruebas conjuntas de las aplicaciones que he hecho en las iteraciones anteriores. Una vez que las pruebas están realizadas y validadas, implantamos la aplicación web en un servidor y subimos la aplicación móvil al Google Play. En esta última iteración acabamos con la documentación y los manuales de las aplicaciones.

En el proceso de desarrollo del proyecto, los diagramas, modelos y archivos que se han ido añadido en las iteraciones están o en esta memoria o en el CD-ROM. Las apreciaciones que hago a lo largo del desarrollo del código lo hago en el propio código.

1.5 Resumen del contenido

En esta sección voy a resumir el contenido de los capítulos que figuran en esta memoria. Es importante remarcar que primero hago la planificación conjunta de las 2 aplicaciones, y después de las etapas de análisis, diseño, implementación y pruebas para cada aplicación, acabo implantando y sacando conclusiones de las aplicaciones. Esta lista de capítulos nos ayudará a tener una idea general de lo que vamos a tratar en proyecto.

- **Capítulo 2: Planificación**

Este capítulo trata sobre la gestión del tiempo que vamos a realizar para llevar a cabo este proyecto. Definiremos las tareas necesarias para su realización, así como la gestión de los riesgos que pueden darse en la ejecución del proyecto. Se planificará de forma conjunta las dos aplicaciones.

- **Capítulo 3: Análisis**

Este capítulo trata sobre el estudio que hacemos sobre el usuario para descubrir sus necesidades e implementarlas en el sistema. Todas las acciones que se van a realizar en la aplicación están reflejadas en los casos de uso. Estos se analizan para la construir la aplicación web desde/para el usuario.

- **Capítulo 4: Diseño**

Este capítulo trata sobre el diseño que se realiza de la aplicación web con respecto al análisis que hemos realizado anteriormente.

- **Capítulo 5: Implementación**

En este capítulo explicamos cómo he implementado la solución para satisfacer las necesidades del usuario, explicando la estructura, los patrones y herramientas utilizadas.

- **Capítulo 6: Pruebas de validación**

En este capítulo exponemos las diferentes pruebas que hago para comprobar que la aplicación funciona correctamente, además de que no he olvidado ningún requisito funcional por hacer.

- **Capítulo 7: Aplicación Móvil**

En este capítulo utilizamos el mismo esquema que la aplicación web, es decir, tiene apartados: Análisis, diseño, implementación y pruebas de validación.

- **Capítulo 8: Conclusiones**

En este capítulo exponemos las conclusiones que he llegado tras realizar el proyecto. También hablo sobre las posibles mejoras que se pueden añadir al proyecto en un futuro.

- **Capítulo 9: Bibliografía**

Este capítulo indicamos todas las fuentes consultadas para realizar este proyecto.

- **Anexo**

Este capítulo muestra los diagramas, hablamos sobre los manuales de la aplicación y de instalación y del contenido del CD-ROM adjunto a esta memoria.

2 Planificación

Este proyecto lo realiza una única persona. No solo se planifica la implementación, sino que también hay que valorar todos y cada uno de los procesos que eso conlleva, como pueden ser las tareas de análisis, diseño, pruebas y las iteraciones que se emplean para llegar al producto final. Con la experiencia adquirida en la universidad, debo ser capaz de estimar los costes temporales, de esfuerzos y económicos que supone realizar este proyecto.

Lo primero que debemos hacer es definir las tareas que supondrá el proyecto, y después valorar los costes, tiempos y esfuerzos que requiere cada actividad para poder planificar adecuadamente. También tengo que tener en cuenta posibles retrasos en las actividades, ya sea por dificultades técnicas o por agentes externos. Esto último, lo tengo en cuenta en el apartado de gestión del riesgo.

Para realizar esta sección hemos consultado el libro UML de Larman. (12)

2.1 Definición de tareas

Las tareas que debo realizar para cada aplicación son:

- A. Análisis de requisitos.
- B. Análisis técnico.
- C. Análisis del sistema.
- D. Diseño del sistema.
- E. Construcción y evaluación de un prototipo.
- F. Implementación del sistema.
- G. Pruebas del sistema.
- H. Instalación del sistema.

Debido a que el sistema completo de este proyecto comprende dos aplicaciones, tengo que diferenciar las distintas tareas de cada aplicación. En este caso las tareas que vamos a realizar son las siguientes:

- A. Análisis de requisitos.
- B. Análisis técnico.
- C. Análisis del sistema.
 - 1. Análisis de la aplicación web.
 - 2. Análisis de la aplicación móvil.
- D. Diseño del sistema.
 - 1. Diseño de la aplicación web.
 - 2. Diseño de la aplicación móvil.
- E. Construcción y evaluación de un prototipo.
 - 1. Construcción y evaluación de un prototipo de la aplicación web.
 - 2. Construcción y evaluación de un prototipo de la aplicación móvil.
- F. Implementación del sistema.

1. Implementación de la aplicación web.
 2. Implementación de la aplicación móvil.
- G. Pruebas del sistema.
- H. Instalación del sistema.

Como vemos algunas tareas son comunes y se pueden realizar para las dos aplicaciones conjuntamente como son la A,B,G y H. Las demás tareas deben de ser tratadas individualmente por cada aplicación.

En la tarea de implementación del sistema, en las dos aplicaciones, damos por supuesto que se van a realizar las pruebas de forma local. Las pruebas del sistema (G), al igual que en la tarea de implementación, se vuelve a comprobar la batería de pruebas que se realiza a nivel local para ver que el sistema funciona correctamente en producción.

2.2 Secuenciación de tareas

La secuencia de las tareas tiene una única línea de trabajo, debido a que hay una sola persona que lo realiza y no puede estar en dos tareas a la vez. El diagrama de secuencia sería el de la figura 1.

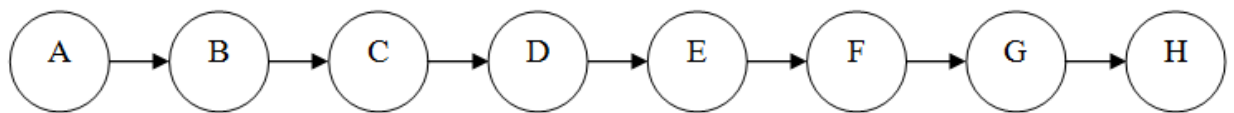


Figura 1 - Planificación de tareas secuencial del sistema ARKivia.

En el caso de que este proyecto lo realizara un equipo de personas, el esquema cambiaría porque algunas de las tareas se realizarían en paralelo, de forma que, lo más seguro, los tiempos se acortarán. El siguiente diagrama de secuencia, de la figura 2, muestra para el caso de que contemos con 2 o más personas.

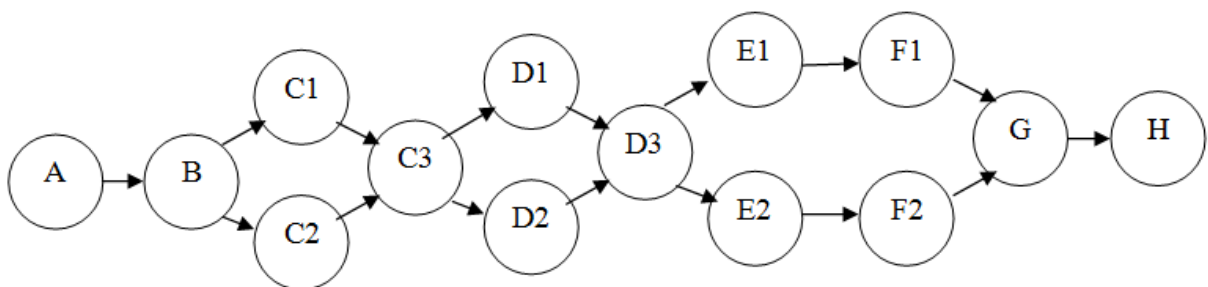


Figura 2 - Planificación de tareas con varias personas trabajando en el sistema ARKivia.

Este proyecto se ha planificado como un híbrido de los dos esquemas anteriores para hacerlo más lógico y simple. Se hace primero un análisis conjunto de funcionalidades básicas de las aplicaciones, y después de presentarlo al cliente, se hace la implementación del resto de funcionalidades. Primero de la aplicación web y después de la aplicación móvil. Para finalizar se prueban las dos aplicaciones y se hace la integración. Al realizar una única persona son tareas secuenciales. El esquema sería de la siguiente forma, representado en la figura 3.

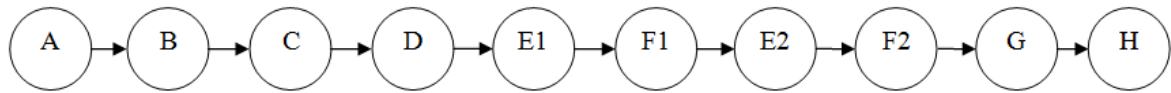


Figura 3 - Planificación final de las tareas del sistema ARKivia.

2.3 Asignación de tiempos

Este apartado recogerá los tiempos esperados de cada tarea para ver cuál sería el tiempo total de nuestro proyecto sin contemplar los riesgos que pueden existir. La fecha de inicio del proyecto es el día 25 de Febrero de 2014 para la fecha de inicio, y como fecha límite 5 de junio de 2014 por la cercanía de los exámenes de junio y julio. Debido a causas ajenas al proyecto, no se pudo realizar dentro de las fechas fijadas, pero si se han cumplido los tiempos predecidos. La siguiente tabla muestra el esfuerzo estimado de cada tarea:

Descripción de la actividad	Actividad	Estimación del esfuerzo de la actividad
Elaboración del análisis de requisitos.	A	24 horas/persona
Elaboración del análisis técnico.	B	24 horas/persona
Elaboración del análisis del sistema.	C	48 horas/persona
Realizar el diseño del sistema.	D	48 horas/persona
Construir y evaluar el prototipo de la aplicación web.	E1	16 horas/persona
Implementar la aplicación web.	F1	60 horas/persona
Construcción y evaluación de prototipo de la aplicación móvil.	E2	4 horas/persona
Implementar aplicación móvil.	F2	48 horas/persona
Definir y ejecutar las pruebas del sistema.	G	8 horas/persona
Instalar el sistema en completo en producción y realizar los manuales.	H	16 horas/persona
Total		288 horas/persona

Esta tabla traduce los días como 4 horas de esfuerzo de una persona en el proyecto, por lo que me da un total de 288 horas/persona para finalizar. Las tareas de análisis y de diseño, C y D, comprenden las horas de esfuerzo que tiene las tareas de construcción de prototipos de la fase 2 a la fase 5. La asignatura de TFG estima unas 300 horas de trabajo que comprende las tutorías con el profesor asignado, realización de la memoria del proyecto, y la presentación. Puedo deducir que es un tiempo razonable, pero hay que tomar en cuenta que pueden surgir problemas durante la realización. Es por ello que hago una gestión del riesgo más adelante.

El diagrama de Gantt me permite ver cómo las tareas están organizadas en la línea del tiempo desde el inicio hasta el fin del proyecto. En el anexo A, ilustración 1, muestro el diagrama con los datos obtenidos anteriormente.

Este análisis me permita saber si estoy a tiempo de realizar el proyecto para la fecha de finalización establecida. Es necesario que se actualice a lo largo del proyecto y haga continuas revisiones de los tiempos que me han llevado las tareas.

Como se puede comprobar estoy en el camino crítico durante todo el tiempo de realización y su puesta en marcha. Esto hace que no tengamos holgura en ninguna tarea, y el retraso de alguna de ellas supone el retraso de todo el proyecto.

Ahora podemos hallar el porcentaje de tiempo de cada iteración respecto del proyecto en su totalidad. En la siguiente tabla vemos las distintas iteraciones, con su porcentaje de tiempos.

	Horas/personas	Días	Porcentaje
Iteración 1	76	19	26,38%
Iteración 2	28	7	9,72%
Iteración 3	72	18	25,00%
Iteración 4	28	7	9,72%
Iteración 5	60	15	20,83%
Iteración 6	24	6	8,33%
Total	288	72	100%

Esta tabla muestra como la primera iteración y las implementaciones final de las aplicaciones ocupan más del 70% del proyecto.

2.4 Gestión del riesgo

Existe la posibilidad que se puedan dar problemas en el desarrollo del proyecto y estos riesgos se deben gestionar para que en el caso de que se den, saber cómo actuar y encontrar una solución rápida y efectiva.

Los riesgos se definen antes de realizar el proyecto para adelantarse a ellos y poder elaborar un plan B en caso de que se manifiesten. Esta gestión exige estar en continua alerta para controlarlos y minimizar el impacto que puedan llegar a tener. Solo se tomarán en cuenta los riesgos de proyecto y técnicos. Existen otros tipos de riesgos como son los costes o los humanos que en este caso no voy a valorarlos. (13; 14)

Se define para cada riesgo los siguientes aspectos:

1. La definición del problema.
2. Consecuencias que puede llevar este riesgo.
3. Medida en la que afecta. (despreciable, marginal, critica, catastrófica)
4. La posible solución.

5. Probabilidad de que ocurra.

Normalmente se buscan los posibles efectos económicos de cada riesgo pero únicamente me centro en el tiempo que se debe añadir a mayores si da el caso. Existen dos apartados que son el de riesgos de proyecto y técnicos.

Las siguientes tablas muestran los posibles casos de riesgo del proyecto:

Definición del problema	Los requisitos iniciales no sean posible en tiempo o en forma para la realización del TFG.
Consecuencias	Aumentar 8 horas/persona o se retrasa 2 días
Grado de afectación	Marginal
Solución	Redefinir los objetivos del proyecto para que si sea viable.
Probabilidad	10%

Definición del problema	Temporada de exámenes.
Consecuencias	Reducción a 1 hora/persona el esfuerzo en esa temporada
Grado de afectación	Critica
Solución	Dejar el proyecto de lado durante la temporada de exámenes y retomarlo cuando termine.
Probabilidad	90%

Definición del problema	Cambios en los requisitos funcionales.
Consecuencias	Añadir 2 horas/persona por cada vez que ocurre este riesgo.
Grado de afectación	Critica
Solución	Adaptar el proyecto a esos nuevos requisitos minimizado el impacto al máximo.
Probabilidad	30%

Definición del problema	Mala definición de las tareas no tomadas en cuenta en la estimación de tiempos.
Consecuencias	Retraso del proyecto.
Grado de afectación	Marginal
Solución	Analizar las tareas y ver los tiempos que se estiman en hacerla. Después, reajustar el proyecto.
Probabilidad	30%

A continuación se muestran las tablas de riesgos técnicos:

Definición del problema	Dificultades a la hora de implementar la tecnología utilizada.
Consecuencias	Retrasar el proyecto o tener que aumentar el esfuerzo por día.
Grado de afectación	Critica
Solución	Dedicar más horas al proyecto para solventar los problemas. Pedir ayuda externa o al tutor para solucionar los problemas que tengo .
Probabilidad	60%

Definición del problema	La versión del framework utilizado dejen de ser actualizado.
Consecuencias	Tener que tomar una decisión al respecto. Aumentar 2 horas/personas.
Grado de afectación	Critica
Solución	Sopesar los pros y los contras para seguir con ese framework o versión.
Probabilidad	20%

Definición del problema	El proyecto utilizado para la realidad aumentada sea más difícil de implementar de lo esperado.
Consecuencias	Retrasar el proyecto.
Grado de afectación	Critica
Solución	Se pedirá ayuda externa si se ve que no vamos a tener el proyecto finalizado para la fecha de entrega.
Probabilidad	75%

Definición del problema	Problemas a la hora de la implantación de módulos.
Consecuencias	Retrasar el proyecto o tener que aumentar 2 horas/persona durante 1 días.
Grado de afectación	Marginal
Solución	Se recopilara mayor información para solucionar los problemas de implantación o se cambiara de servicio para reducir el tiempo invertido en esta tarea.
Probabilidad	40%

En conclusión, vemos que existen riesgos generales y específicos de implementación para este proyecto, que se pueden llegar a dar en mayor o menor grado. Debemos tener presente este plan de gestión a lo largo del proyecto para así tener controlado la estimación de horas para finalizarlo en la fecha deseada. En caso de que se den estos riesgos, tenemos ya definido un plan B pero seguramente no tengamos contemplados todos los riesgos. Es por todo ello, que debemos que extremar la precaución en caso de que se dé un riesgo y si se da, saber solventarlo en el menor tiempo posible.

2.5 Recursos empleados para el proyecto

Para llevar a cabo el proyecto he utilizado las siguientes herramientas:

- Hardware:
 - Ordenador portátil: Intel Core i3 CPU M350 2.27GHZ, 4GB de memoria RAM, 120GB de disco duro, Microsoft Windows 8.1 Professional.
- Software:
 - REM: Esta herramienta me permite hacer un análisis de requisitos funcionales y no funcionales mucho más rápido, generando una plantilla con toda la información necesaria para realizar la tarea de requisitos del sistema con la definición de actores y definición de casos de uso.
 - Astah: Esta aplicación nos permitirá hacer el resto de modelos y de diagramas en UML tanto de análisis como de diseño.
 - Microsoft office 2007: Es una aplicación es de procesamiento de texto. La utilizare para realizar la memoria.
 - Gantt project: Es una herramienta que me permite realizar el diagrama de Gantt.

En los apartados de implementación específico los programas utilizado para su realización.

3 Análisis

Debemos analizar todo lo que rodea el producto para poder dar al usuario lo que necesita. Este análisis, llamado diseño centrado en el usuario, ponemos como protagonista al usuario final del proyecto, para trabajar con él y tener información que me ayuda a definirlo. No podemos hacer un producto sin las especificaciones suyas, que es nuestro usuario final y quien utilizara nuestro producto. Sin un buen análisis del usuario es muy difícil sacar un producto que se adapte a él.

Existen diferentes técnicas centradas en el usuario en la fase de análisis como por ejemplo las entrevistas, los estudios de campo o por ejemplo las sesiones de grupo, donde su principal objetivo es sacar información para saber sus necesidades y poder dar soluciones a sus problemas con nuestro producto.

Debido a que es un proyecto de corta duración debemos limitar este tiempo de análisis de usuario, ya que se puede tardar incluso años. Por esto, lo que se va hacer es trabajar con prototipos viables e ir enseñándoselos al usuario final, que en este caso es el tutor, que nos va añadiendo funcionalidades a la aplicación según van surgiendo necesidades. Se ha trabajado en las tutorías para hacer estudios de que funciones eran necesarias, y cuales ocupaban un lugar secundario.

La recolección de requisitos de la interfaz de usuario, se debe de hacer con los usuarios reales que he sacado información de ellos, en mi caso, el papel de usuario lo hará mi tutor.

3.1 Análisis de usuarios

Cuando se realiza un producto de software, es importante saber cuáles van a ser nuestros clientes, y que características tienen: sexo, edad, nivel académico, ocupación y un largo etc. Dependiendo del proyecto, las particularidades a tener en cuenta cambian y debemos centrarnos únicamente en las que son relevante para este proyecto. Una vez seleccionadas todas las características que nos interesan, debemos encontrar los valores de nuestros clientes para tenerlos en cuenta durante toda la creación del producto y poder satisfacer sus necesidades.

Una vez tenemos la muestra de información de usuario necesaria recogida, debemos de ser capaces de clasificar en grupos estas características y poder encontrar perfiles que se adapten a un usuario común de ese perfil.

Nuestro software gestiona archivos creados por los usuarios que contienen los modelos 3D, así como posibles archivos adyacentes como texturas, animación, imágenes etc. Este tipo de usuario tienen capacidades informáticas altas, ya que es capaz de crear modelos y utilizar herramientas de este tipo.

El otro tipo de usuario encontrado es el que no sabe lo que es la realidad aumentada y los modelos 3D, se encuentre con este proyecto y quiera saber algo más. Para este segundo tipo de personas también tenemos que resolver sus necesidades y hacer que tenga una buena experiencia con nuestro sistema.

Por lo expuesto anteriormente, vamos a definir los dos perfiles que podrían utilizar nuestro proyecto:

	Perfil 1	Perfil 2
Edad	20-50	12-65
Sexo	Ambos	Ambos
Nivel académico	Alto	Medio
Nivel de informática	Alto. Maneja herramientas 3D por lo que estará familiarizado con software de alto nivel que tiene muchas opciones y utiliza internet a diario.	Medio. La sociedad en general en ese rango de edad tiene un nivel medio de informática y experiencia en internet.
Disposición	Tiene ganas de satisfacer sus necesidades por lo que estará probando nuestro sistema para ver si es lo que busca.	Busca el explorar nuevos mundos como es el modelado 3D o la realidad aumentada, si no le gusta lo que ve o es complicado se ira.
Interés	Medio	Bajo

Trabajaremos inicialmente para el perfil 1, dando todas las funcionalidades que necesita. Una vez tengamos bien definido que es lo que desea este perfil, vamos añadiendo más funcionalidades para el perfil 2, para así atraerlos y solventar sus necesidades.

Hay que tener especial cuidado de no hacer una herramienta demasiado técnica, ya que los del perfil 2 quizás no estén cómodos. Lo mismo ocurre si queremos hacer un software demasiado informativo, entonces los del perfil 1 puede que no encuentren soluciones a sus necesidades en la aplicación. Tampoco debemos caer en la tentación de querer poner todas las funcionalidades de uno y de otro. Valoramos la virtud que supone estar en el punto medio.

3.2 Definición de los personajes

Una vez definidos los perfiles, vamos a materializarlos y meternos en el papel del personaje que va utilizar nuestro sistema.

Hemos hallado 2 perfiles, por lo que vamos a definir 4 personajes que puedan ser perfectamente usuarios de nuestra aplicación:

Perfil 1

Alba: 34 años, directora de diseño en una empresa de videojuegos, experta en modelado 3D, tiene una gran cantidad de modelos y su principal problema es su desorganización. Utilizará nuestra aplicación para tener unificados todos los modelos con los que trabaja.

Tomás: 26 años, grado superior en diseño gráfico, su hobby es hacer modelos 3D los fines de semana, le encanta intercambiar sus modelos con otras personas, solo conoce a 2 personas de su entorno que modelen. Utilizara nuestra aplicación compartir modelos con otros.

Perfil 2

Marcos: 54 años, electricista, no sabe lo que es la realidad aumentada ni los modelos 3D. Utilizara nuestra aplicación para informarse de que son los modelos 3D y la realidad aumentada. Quiere jugar con su hijo enseñándole modelos 3D en el móvil.

Óscar: 18 años, sin trabajo ni estudios, le gusta los videojuegos y quiere aprender a hacer modelos 3D. Utiliza ARkivia para ver modelos de los demás y modificarles.

3.3 Análisis de tareas

Para tener un orden de prioridad voy a definir las tareas necesarias para englobar la totalidad de las funciones que requiere la aplicación.

Tarea 1 - Gestionar modelos: Las funcionalidades de dar de alta, baja y poder modificar los modelo 3D.

Tarea 2 - Gestionar álbum: Las funcionalidades de dar de alta, baja y poder modificar de álbum. Esta tarea sirve para organizar los modelos de los usuarios.

Tarea 3 - Gestionar complementos: Las funcionalidades de dar de alta, baja y poder modificar los complementos de los modelos. Esto servirá para poder organizar no solo los modelos, sino también los archivos relacionados con el modelo.

Tarea 4 - Intercambios de modelos: Las funcionalidades necesarias de configuración, se encuentran en el área de usuario.

A mayores de estas tareas necesitamos:

Tarea 5 - Gestionar de usuario: Las funcionalidades de organiza los usuarios de la aplicación para poder dar acceso a un área de usuario.

Nuevas funcionalidades asignadas por el tutor después del primer prototipo:

Tarea 6 - Modelo pendiente: La funcionalidad de hacer una petición pendiente para tener los datos ya introducidos. Por ejemplo, pueden introducir el nombre descripción el álbum, etc.

Tarea 7 - Sistema de log: La funcionalidad de guardar los registros de la aplicación.

Tarea 8 - Descarga de los archivos: La funcionalidad de dar la opción al usuario para poder descargarse los modelos subidos al sistema.

3.4 Modelo de dominio

Este modelo es muy importante, ya que representa las relaciones existentes entre las entidades del sistema. Las entidades escenifican, en nuestro sistema, elementos de la vida real. Las relaciones en el modelo explican el comportamiento que tienen las entidades entre ellas. Un buen modelo de dominio, nos ayudará a entender la realidad logrando que nuestra aplicación sea capaz de trabajar de forma similar.

Tras haber definido las tareas y encontrado las entidades principales nos disponemos a realizar el modelo de dominio, éste consiste en relacionar entidades. Si dos entidades están relacionadas debemos especificar que tipo de relación las une.

En mi caso, la figura 4 representa el modelo de dominio para la aplicación.

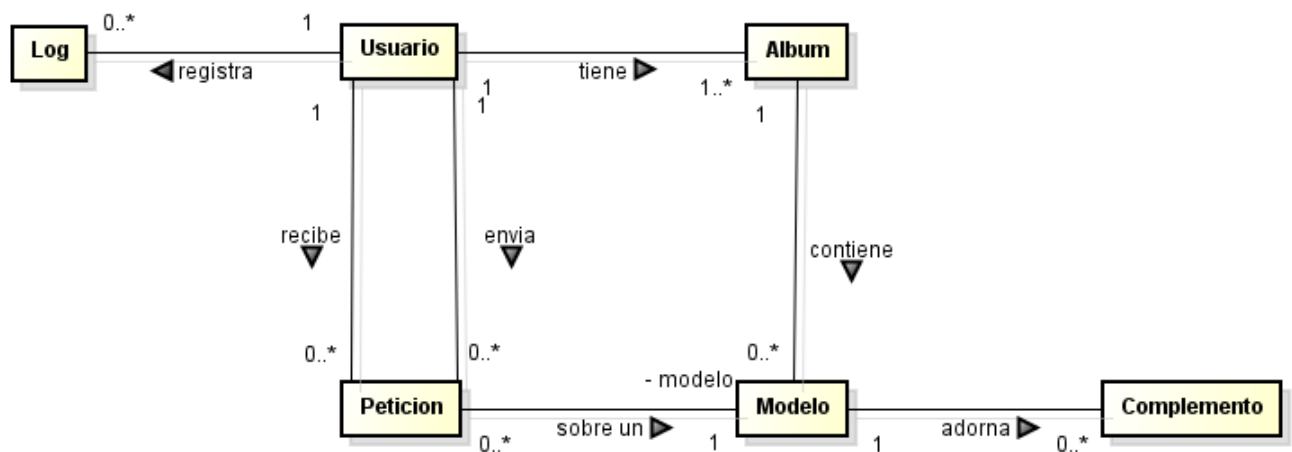


Figura 4 - Modelo de dominio de ARkivia web.

Como podemos ver todas las entidades son representadas: modelo, álbum, complemento, peticiones, usuario y log.

Las relaciones de cada una son:

1. modelo - álbum: un modelo es de un único álbum y en un álbum puede tener modelos.
2. modelo - complemento: un modelo puede tener complemento y un complemento es de un modelo.
3. álbum - usuario: un álbum es de un usuario y un usuario puede tener varios álbumes.
4. usuario - petición: un usuario puede hacer peticiones y una petición es hecha por un usuario.

5. petición - modelo: una petición se hace sobre un modelo y un modelo puede tener varias peticiones.
6. log - modelo: un usuario puede registrar un log y un log solo es registrado por un usuario.

Esta son las relaciones que existen en el modelo de domino. No existe gran variedad de tipo de relaciones, únicamente las N-1, pero también existen las N-M o 1-1 que no aparecen en este proyecto.

3.5 Requisitos funcionales

En este apartado definiremos el comportamiento del sistema. Por cada requisito funcional le pondremos una referencia para poder referirnos a ella más adelante en el manual.

El sistema deberá ofrecer las siguientes funcionalidades:

- RF-1: Almacenar los modelos de los usuarios.
- RF-2: Editar el modelo que está en el sistema.
- RF-3: Borrar los modelo del sistema si el usuario lo desea.
- RF-4: Descargar modelo.
- RF-5: Donar modelo.
- RF-6: Cambiar modelo de álbum.
- RF-7: Buscar modelo.
- RF-8: Añadir modelo desde galería.
- RF-9: Insertar álbum.
- RF-10: Editar álbum.
- RF-11: Suprimir álbum.
- RF-12: Descargar un álbum.
- RF-13: Descargar todos los álbum.
- RF-14: Insertar complemento.
- RF-15: Editar complemento.
- RF-16: Suprimir complemento.
- RF-17: Descargar complemento.
- RF-18: Petición de un modelo.
- RF-19: Petición de un modelo, rellenando los datos previamente.
- RF-20: Rechazar una petición.
- RF-21: Aceptar una petición.
- RF-22: Editar los datos del usuario.
- RF-23: Registrarse en la aplicación.
- RF-24: Poder iniciar y cerrar sesión con un usuario .
- RF-25: Darse de baja en el sistema.

3.6 Requisitos no funcionales

Este sistema tiene unos objetivos generales claros que hemos expuesto anteriormente.

Trataremos puntos como la rapidez del usuario en aprender a usar las aplicaciones, los tiempos estimados de ciertas actividades claves para los diferentes tipos de usuario.

Vamos a dividir en dos esta sección para tratar los dos perfiles por separado.

El perfil 1, recordemos, es un modelador 3D con un nivel alto de manejo de informática de usuario. Para él es primordial la disponibilidad de sus archivos, y que estos estén en un lugar seguro y disponible. Como requisito no funcional, la acción de subir un archivo por primera vez a ARkivia debe de ser de menos de 2 minutos y para posteriores veces, menos de 1 minuto. Para encontrar un archivo organizado en álbum tardará menos de 30 segundos. Los errores en estas actividades deben de ser menores del 5%, ya que son tareas sencillas. Como medida de seguridad los archivos deben de estar alojados en servidor o servidores que ofrezcan caídas de menos de 24 horas al año.

Para los usuarios que se adaptan al perfil 2, aquellos que no tienen una experiencia en modelos 3D y su objetivo es saciar su curiosidad, los tiempos van a ser más elevados que el párrafo anterior, ya que su manejo de internet es quizás menos hábil. Para insertar su primer modelo 3D tardará menos de 4 minutos. Es un tiempo elevado, ya que debe encontrar un modelo 3D inicial que le ofreceremos nosotros, si lo quiere, y además ingresarle en un formulario que tal vez sea desconocido para él. Si no es su primera vez, estimamos un tiempo de menos de 2 minutos. En el caso de realizar una búsqueda de objeto en la galería, no debe de ser mayor que de 1 minutos hasta encontrar su objetivo. Las tasas de error en este tipo de usuario no deben de ser mayor del 10%.

Las peticiones de modelos 3D a otros usuarios, para los dos tipos de usuarios, debe de ser, una vez encontrado el objeto que desean, de menos de 25 segundos.

Para todas estas tareas, siempre partimos de la página principal después de haberse autenticado.

El registro en ARkivia no debe de durar más de 1 minuto y 20 segundos para los dos tipos de usuarios, incluyendo la activación por email de la cuenta.

3.7 Casos de uso

Una vez analizados los requisitos funcionales y no funcionales, realizo los diagramas de casos de uso del sistema. En este sistema lo voy a organizar por paquetes y cada paquete tiene su caso de uso. El diagrama general de paquetes es la figura 5.

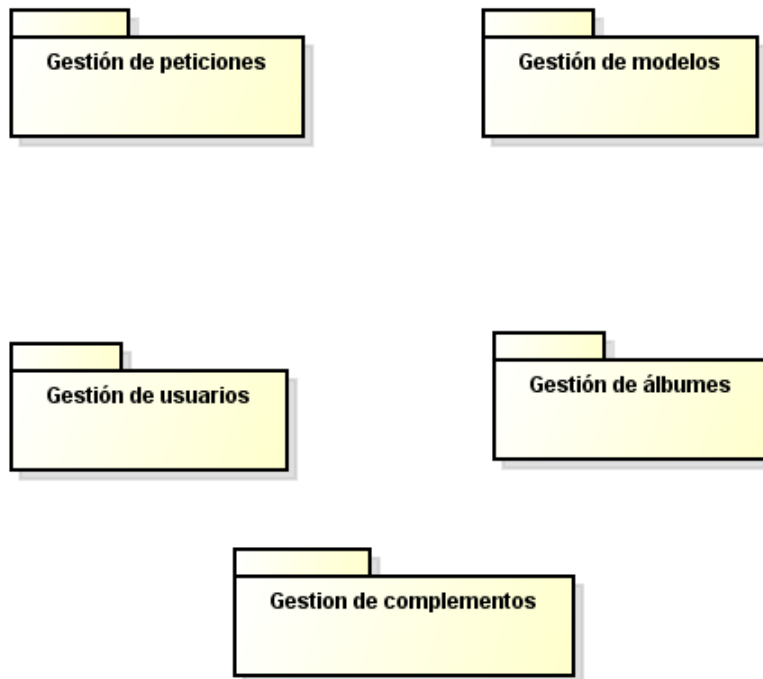


Figura 5 - Paquetes de casos de uso de ARKivia web.

En las siguientes figuras 6, 7, 8, 9 y 10 muestro los casos de uso de cada paquete.

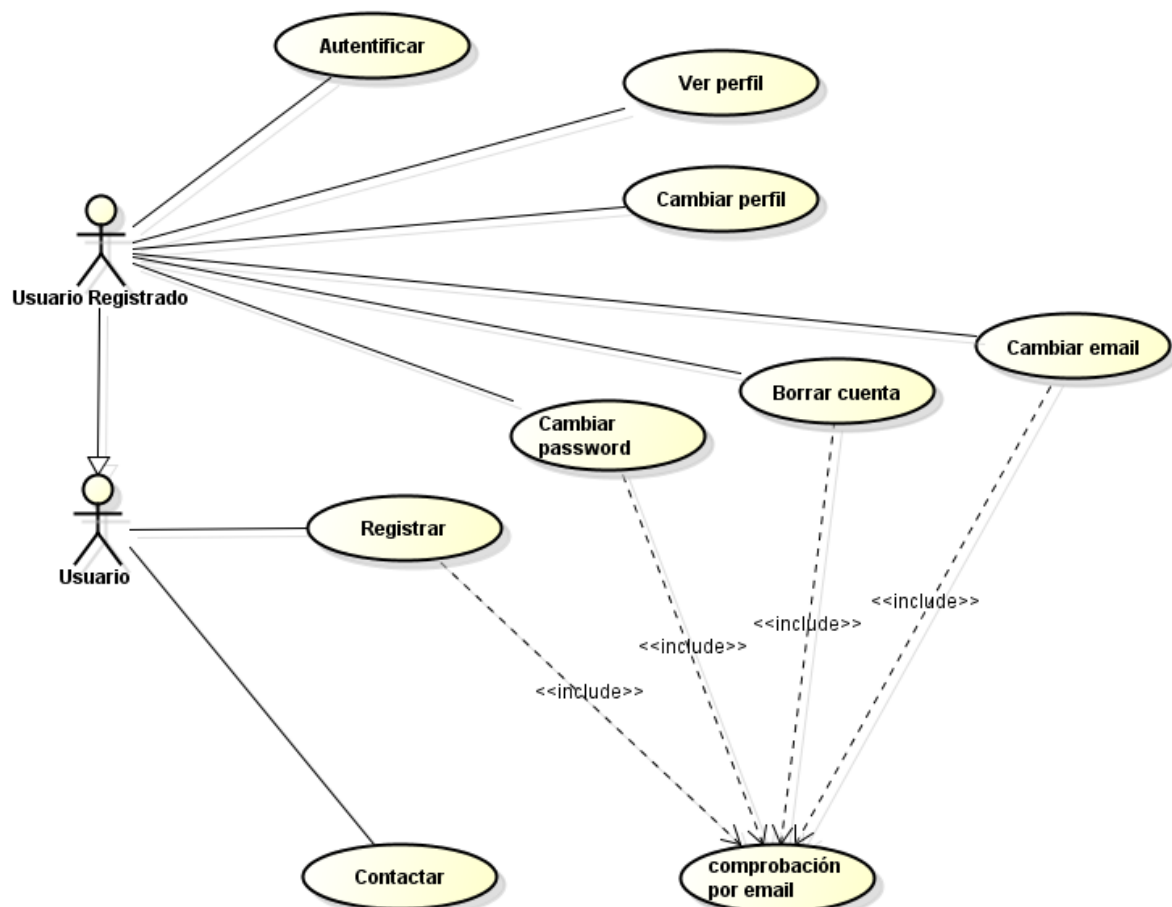


Figura 6 - Diagramas de casos de uso de gestión de usuarios.

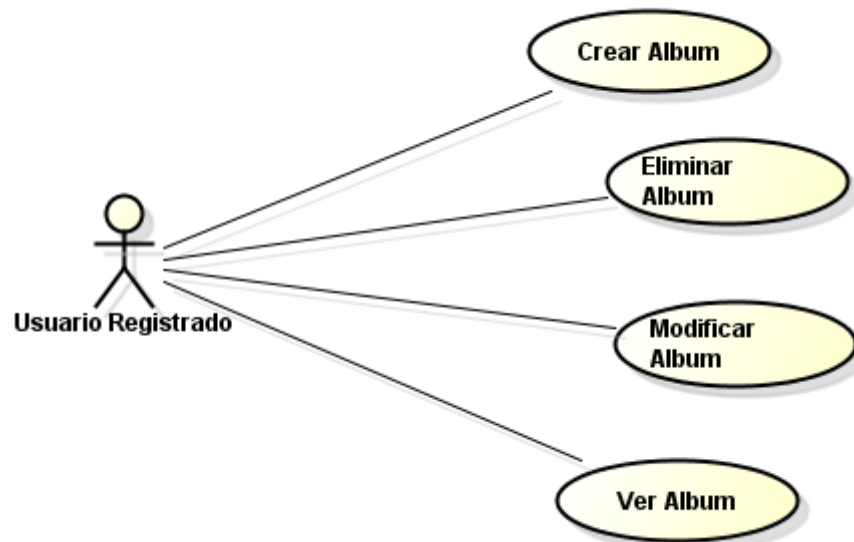


Figura 7 - Diagramas de casos de uso de gestión de álbum.

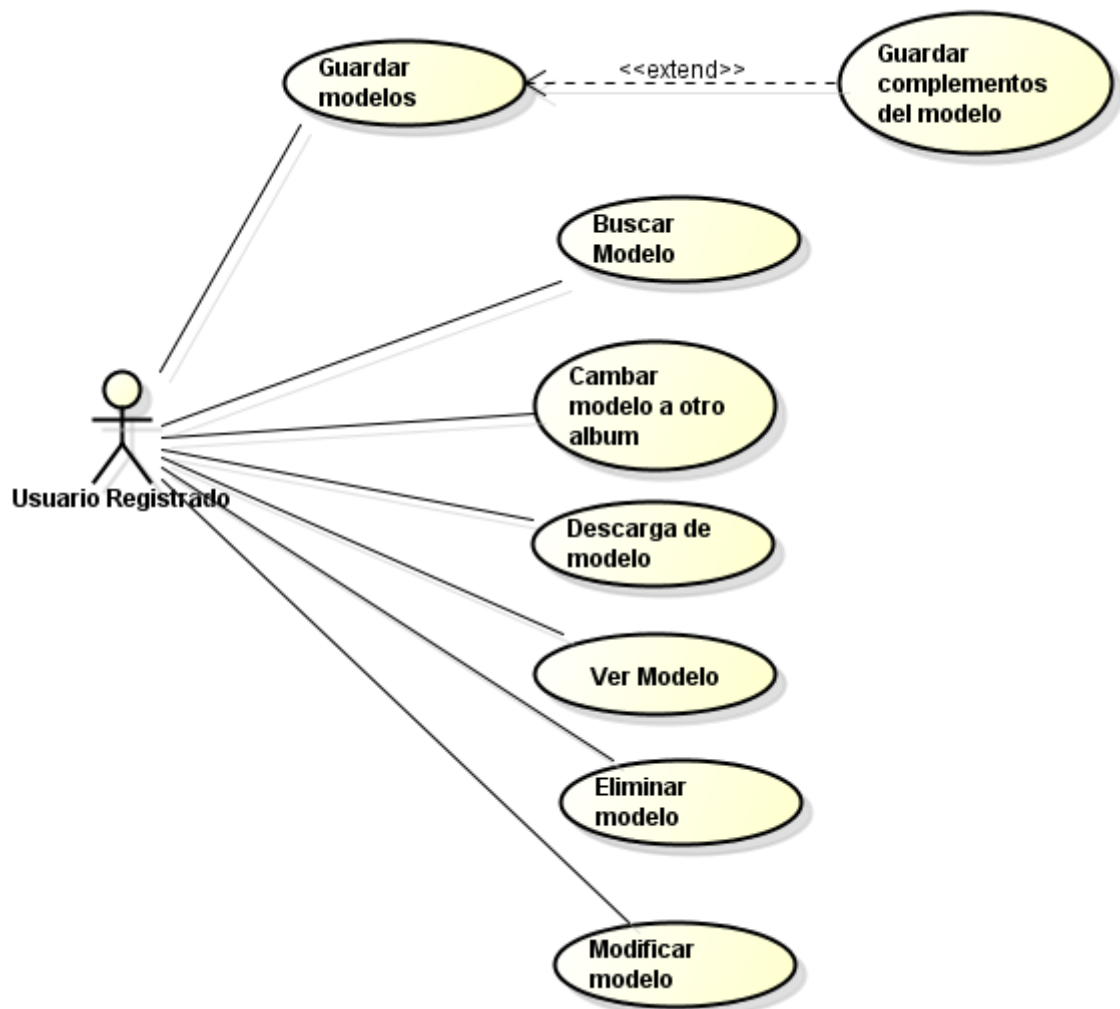


Figura 8 - Diagramas de casos de uso de gestión de modelo.

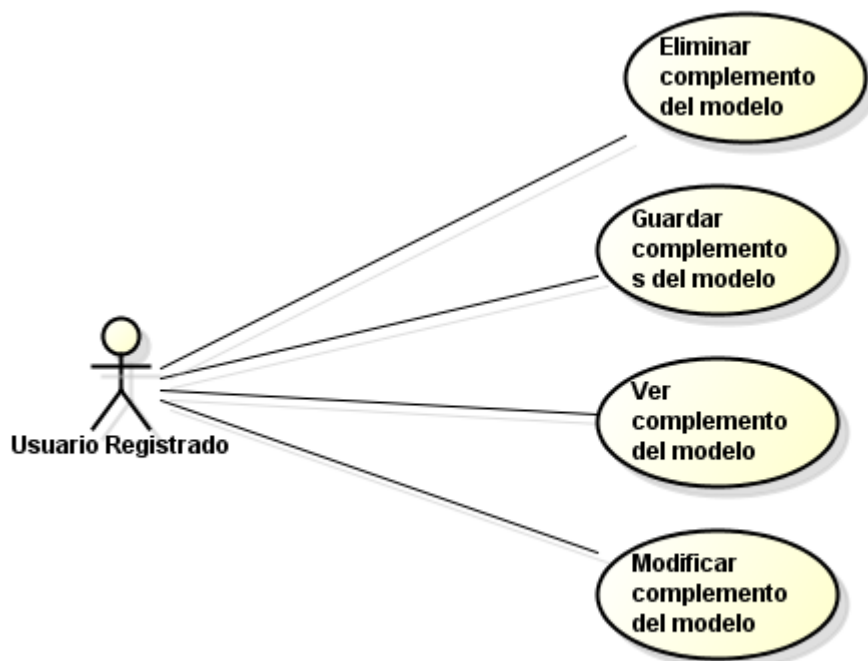


Figura 9 - Diagramas de casos de uso de gestión de complemento.

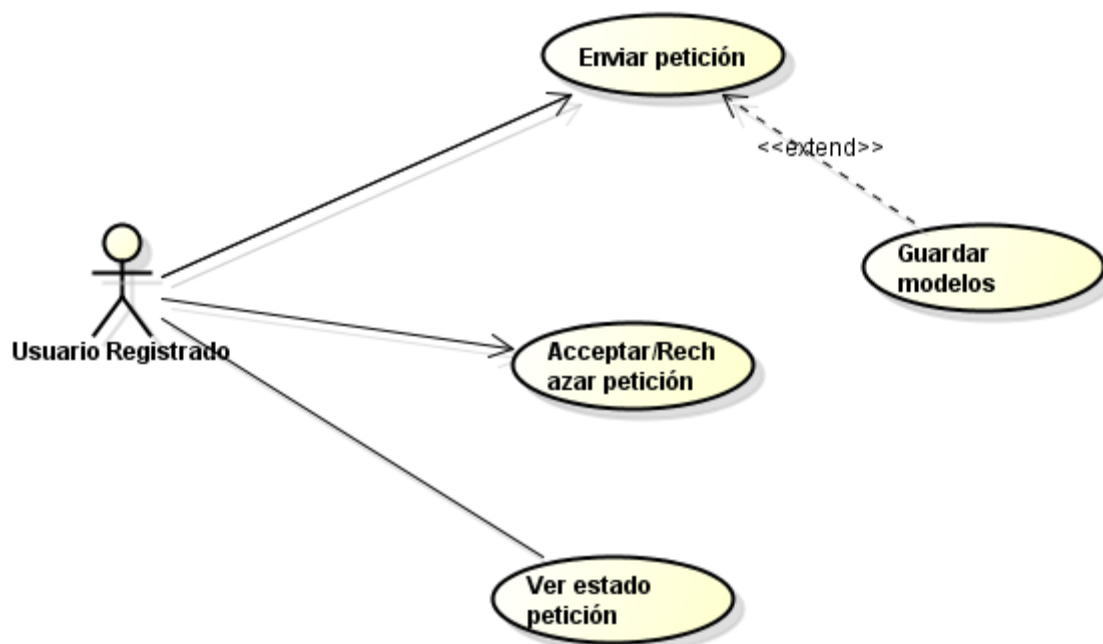


Figura 10 - Diagramas de casos de uso de gestión de petición.

Los actores son la representación de los usuarios que van a utilizar nuestra aplicación. Los que realizan las acciones en el sistema son ellos, y ocurre que no todos pueden hacer todas las acciones. Es la razón por la cual se definen actores y podemos hacer un análisis de ellos, con la que delimitar las funciones de cada uno. Esto nos facilita entender el sistema y que pueden hacer cada actor delimitando que es lo que pueden hacer unos y otros.

En este sistema podemos definir dos tipos de actores, aunque el segundo pueda tener las mismas funciones que el primero. En las tablas que están a continuación lo veremos con más detalle.

ACT-1	Usuario
Descripción	Este actor representa los usuarios que no se logueado, ya sea porque no se han registrado o no han confirmado el email.

ACT-2	Usuario registrado
Descripción	Este actor representa los usuarios que se ha registrado en el sistema y han realizado la activación por email. Además se han logueado en el sistema.

Cada casos de uso definimos nombre, descripción, entrada, precondiciones y postcondiciones. Algunos casos de uso no se tomaron en cuenta debido a que el actor por la simple navegación por la aplicación web se realizan solos.

Los casos de uso que vamos a definir son los siguientes:

CU-1	Registrar
Descripción	El actor ACT-1 se registra en el formulario para estar en el sistema de ARkivia.
Entrada	<ul style="list-style-type: none"> • Nick: "Sandra22" • Email: sandra@gmail.com • Password: Paris21 • Repet password: Paris21
Precondiciones	El actor ha llegado al formulario de registrar.
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos. 2. El Sistema introduce los datos en la base de dato y envía un email al correo para pedir confirmación. 3. CU-2 4. El sistema muestra un mensaje como se ha acabado el proceso de registro.
Postcondiciones	Ninguna
Excepciones	<ol style="list-style-type: none"> 1. Algunos de los datos son erróneos 1. Faltan campos obligatorios por introducir
Nota	Ninguna

CU-2	Comprobación por email.
Descripción	El actor ACT-1 ha recibido el correo de confirmación y debe confirmarlo.
Entrada	Ninguna
Precondiciones	CU-1: llegar al paso 5.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace click en el link del correo para activar la cuenta. 2. El sistema modifica el estado del registro en la base de datos.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

CU-3	Contactar
Descripción	El actor ACT-1 o ACT-2 enviar un comentario, sugerencia o duda al email del administrador de ARKivia.
Entrada	<ul style="list-style-type: none"> Nombre: Sandra Email: sandra@gmail.com Asuntos: Problema con el registro Texto: Me sale error 404. ¿Qué puedo hacer?
Precondiciones	Llegar al paso
Secuencia	<ol style="list-style-type: none"> El actor introduce los datos y da al botón de enviar. El sistema envía un email al administrador de ARKivia para que este le conteste.
Postcondiciones	El sistema muestra un mensaje para informar que se ha enviado correctamente.
Excepciones	<ol style="list-style-type: none"> Algunos de los datos son erróneos. Faltan campos obligatorios por introducir.
Nota	Ninguna

CU-4	Autenticar.
Descripción	El actor ACT-1 ha recibido el correo de confirmación y debe confirmarlo que es realmente quien dice ser..
Entrada	Ninguna
Precondiciones	CU-1 completado correctamente.
Secuencia	<ol style="list-style-type: none"> El actor hace click en el link del correo para activar la cuenta. El sistema modifica el estado del registro en la base de datos.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	En este caso de uso el ACT-1 pasa a ser ACT-2.

CU-5	Añadir modelo
Descripción	El actor ACT-2 guarda un modelo en el sistema de ARKivia.
Entrada	<ul style="list-style-type: none"> Elegir album: Nuevo Nombre: Dragón Archivo obj: tamaño - 462KB Descripción: Dragón enorme Privado: Activado Imagen: tamaño - 4KB
Precondiciones	Se ha llegado al formulario de ingresar. modelo.
Secuencia	<ol style="list-style-type: none"> El actor introduce los datos de entrada y hace click en el botón de guardar. El sistema guarda el modelo y crea el registro del modelo en la base de datos.
Postcondiciones	Se muestra la información del nuevo modelo introducido.
Excepciones	<ol style="list-style-type: none"> Algunos de los datos son erróneos. Faltan campos obligatorios por introducir.
Nota	Ninguna

CU-6	Editar modelo
Descripción	El actor ACT-2 quiere editar un modelo que ya tiene en ARkivia
Entrada	<ul style="list-style-type: none"> • Nombre: Dragón 2.0 • Archivo obj: tamaño - 872KB • Descripción: Dragón
Precondiciones	El sistema está mostrando al actor el formulario de editar modelos.
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos de entrada y hace click en guardar. 2. El sistema modifica el registro del modelo en la base de datos.
Postcondiciones	El sistema muestra la vista del modelo con los datos actualizados
Excepciones	1. Algunos de los datos son erróneos.
Nota	Ninguna

CU-7	Eliminar modelo
Descripción	El actor ACT-2 borra el modelo de su cuenta
Entrada	Ninguna
Precondiciones	Ninguna
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en el link en el link de borrar modelo. 2. El sistema muestra un mensaje al cliente de si está seguro. 3. El actor hace clic en el botón de borrar. 4. El sistema borra el modelo del servidor y elimina el registro del modelo de la base de datos.
Postcondiciones	El sistema muestra el álbum donde estaba el modelo.
Excepciones	3. El actor vuelve atrás
Nota	Ninguna

CU-8	Descargar modelo
Descripción	El actor ACT-2 quiere descargarse un modelo de la aplicación web.
Entrada	Ninguna
Precondiciones	Ninguna
Secuencia	<ol style="list-style-type: none"> 1. El actor hace click en el boton de descargar modelo. 2. El sistema reagrupa en una carpeta todos los complementos, si los tiene, y el modelo y lo comprime.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Este mecanismo se utiliza también para todas las descargas: Todo los álbumes y su contenido, un álbum y un complemento

CU-9	Cambiar el modelo de álbum
Descripción	El actor ACT-2 ha recibido el correo de confirmación y debe confirmarlo.
Entrada	Ninguna
Precondiciones	Existan al menos dos álbumes. El sistema muestra el formulario de cambiar de álbum al modelo
Secuencia	<ol style="list-style-type: none"> 1. El actor selecciona el álbum nuevo y da a guardar. 2. El sistema cambiar el registro del modelo asignándole el nuevo álbum.
Postcondiciones	El sistema muestra el álbum donde está el modelo que se ha cambiado.
Excepciones	Ninguna
Nota	Ninguna

CU-10	Buscar modelo
Descripción	El actor ACT-2 busca un modelo en la galería.
Entrada	<ul style="list-style-type: none"> • Buscar: dragon
Precondiciones	Ninguno
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos de entrada y pulsa el botón buscar. 2. El sistema encuentra los modelos que tengan como título o categoría que tengan la búsqueda.
Postcondiciones	El sistema muestra por pantalla los modelos que coinciden.
Excepciones	Ninguna
Nota	Ninguna

CU-11	Donar modelo.
Descripción	El actor ACT-2 desea donar un modelo a ARkivia.
Entrada	Ninguna
Precondiciones	El sistema muestra la vista del modelo seleccionado.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en el botón de donar modelo. 2. El sistema duplica el modelo y se lo da usuario ARkivia.
Postcondiciones	El sistema muestra un mensaje de agradecimiento
Excepciones	Ninguna
Nota	El usuario de ARkivia es un usuario especial que se crea al instalar la aplicación. En el apartado de diseño de la memoria hablo más sobre este caso de uso.

CU-12	Guarda complemento del modelo
Descripción	El actor ACT-2 ha recibido el correo de confirmación y debe confirmarlo.
Entrada	<ul style="list-style-type: none"> • Nombre: Dibujo • Descripción: Dibujo del modelo v1.0 • Complemento: tamaño - 163KB
Precondiciones	Se realice al menos un caso de uso CU-5. El sistema nos muestra el formulario de complementos tras haber seleccionado un modelo.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en el link del correo para activar la cuenta. 2. El sistema modifica el estado del registro en la base de datos.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

CU-13	Editar complemento
Descripción	El actor ACT-2 ha recibido el correo de confirmación y debe confirmarlo.
Entrada	<ul style="list-style-type: none"> • Nombre: Dibujo • Descripción: Dibujo del modelo v1.0
Precondiciones	CU-12
Secuencia	<ol style="list-style-type: none"> 1. El actor hace click en el link del correo para activar la cuenta. 2. El sistema modifica el estado del registro en la base de datos.
Postcondiciones	El sistema muestra la información del complemento.
Excepciones	Ninguna
Nota	Ninguna

CU-14	Suprimir complemento
Descripción	El actor ACT-2 quiere borrar un complemento de un modelo.
Entrada	Ninguna
Precondiciones	CU-12
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en el botón de borrar complemento. 2. El sistema borra el registro en la base de datos y el archivo del complemento.
Postcondiciones	El sistema muestra la información del modelo.
Excepciones	Ninguna
Nota	Ninguna

CU-15	Guarda álbum
Descripción	El actor ACT-1 crea un nuevo álbum.
Entrada	<ul style="list-style-type: none"> • Nombre: Proyecto VEO • Descripción: Modelos del proyecto VEO • Imagen: tamaño - 63KB
Precondiciones	El sistema muestra el formulario de crear un álbum nuevo.
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos y hace clic en guardar álbum. 2. El sistema crea un registro en la base de datos y guarda la imagen en el servidor.
Postcondiciones	El sistema muestra la vista de los álbumes
Excepciones	Ninguna
Nota	Ninguna

CU-16	Editar álbum
Descripción	El actor ACT-2 ha recibido el correo de confirmación y debe confirmarlo.
Entrada	<ul style="list-style-type: none"> • Nombre: Dibujo • Descripción: Dibujo del modelo v1.0
Precondiciones	CU-15
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos y hace clic en el botón de guarda. 2. El sistema modifica los datos del registro en la base de datos.
Postcondiciones	El sistema muestra la información del álbum.
Excepciones	Ninguna
Nota	Ninguna

CU-17	Suprimir álbum
Descripción	El actor ACT-2 quiere borrar un álbum.
Entrada	Ninguna
Precondiciones	CU-15
Secuencia	<ol style="list-style-type: none"> 1. El actor hace click en el botón de borrar álbum. 2. El sistema pregunta si quiere guardar los modelos que contiene. 3. [Alternativa 1] El actor pulsa si quiere guardar los modelos. [Alternativa 1] El sistema modifica los registros de los modelos y se asignan el álbum que se llama "Sin clasificar" y borra el álbum de la base de datos. 4. [Alternativa 2] El actor pulsa no quiere guardar los modelos. [Alternativa 2] El sistema borra todos los modelos que contiene el álbum y además el álbum.
Postcondiciones	El sistema muestra la vista de álbumes.
Excepciones	Ninguna
Nota	Ninguna

CU-18	Enviar petición de modelo
Descripción	El actor ACT-2 quiere un modelos de otro usuario de ARkivia y se lo pide enviando una petición.
Entrada	<ul style="list-style-type: none"> Comentario: Hola, quería utilizar tu modelo para mi proyecto. [Opcional, introducir modelo pendiente] <ul style="list-style-type: none"> Nombre: televisor Privado: activado
Precondiciones	El ACT-2 selecciona un modelo de la galeria y el sistema muestra el formulario de envío de peticiones.
Secuencia	<ol style="list-style-type: none"> El actor introduce los datos de entrada y hace click en el botón de enviar petición. [Opcional] También introduce los datos del modelo pendiente. El sistema registra una petición al usuario del modelo. [Opcional] Además el sistema introduce los datos en un registro de modelo con estado pendiente.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

CU-19	Aceptar petición
Descripción	El actor ACT-2 ha decidido aceptar la petición y ceder el modelo que le han pedido.
Entrada	Ninguna
Precondiciones	Otro usuario haya realizado el caso de uso CU-18 de uno de sus modelos. El sistema muestra la lista de peticiones pendientes.
Secuencia	<ol style="list-style-type: none"> El actor hace click en el botón de aceptar. El sistema modifica el estado del registro de la petición en la base de datos.
Postcondiciones	El sistema redirección a la lista de peticiones.
Excepciones	Ninguna
Nota	Si introdujo el modelo estará en el álbum sin clasificar con los datos introducidos. Si no introdujo los datos se le asigna por defecto un nombre y se registra en el álbum sin clasificar.

CU-20	Rechazar petición
Descripción	El actor ACT-2 ha decidido rechazar la petición.
Entrada	Ninguna
Precondiciones	Otro usuario haya realizado el caso de uso CU-18 de uno de sus modelos. El sistema muestra la lista de peticiones pendientes.
Secuencia	<ol style="list-style-type: none"> El actor hace click en el botón de rechazar. El sistema modifica el estado del registro en la base de datos.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Se debe eliminar la copia del modelo cuando se ha realizado la petición

CU-21	Editar perfil
Descripción	El actor ACT-2 quiere rellenar o editar algún dato de su perfil.
Entrada	<ul style="list-style-type: none"> Nombre: Luis Apellidos: Pérez García
Precondiciones	El sistema se encuentra en la parte de editar perfil , mostrando el formulario.
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce los datos y hace clic en el botón de guardar. 2. El sistema modifica registro en la base de datos del perfil de usuario.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

CU-22	Cambiar el email
Descripción	El actor ACT-2 quiere cambiar el email de su usuario.
Entrada	<ul style="list-style-type: none"> Email nuevo: antonio@gmail.com Contraseña: isengard24
Precondiciones	El sistema se encuentra en la vista de cambiar email, mostrando el formulario.
Secuencia	<ol style="list-style-type: none"> 1. El actor introduce lo datos de entrada y da al botón de guardar. 2. El sistema envía un email al usuario para la confirmación.
Postcondiciones	CU-2
Excepciones	Ninguna
Nota	Una vez activado ya puede loguearse con el nuevo email la información emitida por ARkivia sera enviada a ese email.

CU-23	Cambiar el contraseña
Descripción	El actor ACT-2 quiere cambiar la contraseña de su usuario.
Entrada	<ul style="list-style-type: none"> Contraseña nuevo: isengard20 Contraseña: isengard24
Precondiciones	El sistema se encuentra en la vista de cambiar contraseña, mostrando el formulario.
Secuencia	<p>El actor introduce lo datos de entrada y da al botón de guardar.</p> <p>El sistema envía un email al usuario para la confirmación.</p> <p>CU-2</p>
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Una vez activado ya puede loguearse con la nueva contraseña.

CU-24	Borrar cuenta
Descripción	El actor ACT-2 quiere borrar la cuenta.
Entrada	Ninguna
Precondiciones	El sistema se encuentra mostrando la vista de borrar cuenta.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en borrar cuenta. 2. El sistema borra todos los registros vinculados a este usuario así como los modelos guardados en el servidor.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Este caso de uso se dividió en 3 partes más adelante pudiendo elegir si los modelos, todos o solo los públicos, podía ser donados a ARKivia o borrar completamente los modelos sin donarles.

Una de las implementaciones complementarias, que no se puede expresar en caso de uso, es la implementación de un registro de log que va guardando, en la base de datos, toda la traza de navegabilidad que van realizando los usuarios por la aplicación. Este añadido nos permite 2 cosas, la primera es que vamos a poder analizar las secciones que más se visitan, y podremos saber que es lo que funciona dentro de la aplicación y lo que no. Y la segunda, es para el usuario, podremos ofrecerle un ranking de los modelos más vistos, más utilizados y más famosos.

Una característica que se ha añadido también por petición expresa del cliente ha sido la utilización de un “bread crum”, que permite al usuario volver a las páginas visitada anteriormente. Esto mejorará la experiencia del usuario.

Otra característica adicional es la creación de una ayuda más dinámica, además de la creación de FAQs y de ofrecer los manuales en formato PDF. Ésta característica permite ir a la sección cuando tengas una duda y poder resolverla con una pequeña descripción.

Por último, se ha incluido un visualizador 3D en la propia página para los modelos con formato.obj. Esto permite al usuario visualizar en 3D el modelo que tenemos en ARKivia.

Estas 4 últimas características se han incorporado al final del prototipo y se valorarán como casos de uso especiales, ya que no tiene una función con el dominio en si, sino que proporciona una funcionalidad extra.

En conclusión, decir que estos son los casos de uso que se han encontrado en el análisis, con una primera intervención del tutor, añadiendo ciertas funcionalidades. Seguramente estos casos de uso sufran alguna modificación para satisfacer necesidades o bien por temas de usabilidad. El análisis de una aplicación tiene vida y evolucionará hasta llegar a un producto consolidado.

4 Diseño

El diseño de cualquier sistema informático, es la fase más importante en el creación del software. Un error en esta etapa podría tener consecuencias negativas en el resto del proyecto. Es por ello que dedicamos gran parte del tiempo en esta etapa, y de ese modo asegurarnos hacer una aplicación de gran calidad.

Esta fase la hemos dividido en 4 partes. En la primera hablaremos sobre las elecciones de la tecnología que voy a utilizar. En la siguiente parte, expongo la arquitectura utilizada y los diferentes patrones definiendo el diagrama de clases y más concretamente el MVC. En la tercera parte, diseño la base de datos. Y para finalizar, muestro la navegación de la aplicación web.

4.1 Tecnología seleccionada

Para la creación de la aplicación web, lo primero que debemos hacer es decidir el framework que voy a utilizar para realizarlo.

Los CMS, que son las siglas en inglés de *Content Management System*, hacen referencia a los programas que nos permiten crear una estructura de soporte para la creación de aplicaciones web. Su principal objetivo es la rapidez con la que se crea contenido y su gran funcionalidad en épocas tempranas de su desarrollo. Este tipo de programas está muy expandido y las empresas están muy interesadas en que se desarrollen, ya que ofrecen un marco muy apetecible por su rapidez y eficacia.

Por otra parte, tenemos los frameworks, que son una estructura de librerías y funciones que nos facilitarán la tarea de desarrollar un proyecto web. Esta estructura, puede estar escrita en numerosos lenguajes de programación. (15; 16)

Actualmente en el mercado existen diferentes CMS y frameworks capaces de realizar este proyecto. Debo evitar elegir uno demasiado versátil o que pueda restringir ciertas partes del proyecto, ya que se debe adaptar tanto en tiempo, como en requisitos del proyecto. El lenguaje de programación que se utilizará, no será una variable a tomar en cuenta para la elección de un CMS.

Las diferentes plataformas que barajo, ordenadas de mayor a menor resultado en Google, son:

- Spring: Este CMS está programando en el lenguaje JAVA, por lo que tienen una gran comunidad que lo respalda. Compite directamente con EJE, que también hace parte de la familia de este lenguaje de programación. Se construye añadiendo módulos prefabricados para agilizar el proceso de producción, y debe de tener el usuario final la máquina virtual java. Utiliza una programación alternativa que es la inversión de control. Aproximadamente 25.400.000 resultados en Google. (17)
- Joomla: Este sistema de gestión de contenidos, además de asemejarse en características al anterior, se programa en PHP y se construye a base de añadir módulos. Se llega por la tercera versión y tiene más de 7700 extensiones actualmente. Aproximadamente 13.300.000 resultados en Google. (18)

- ASP.NET: Este Framework utiliza el lenguaje C# y utiliza MVC como arquitectura de diseño. Tiene un punto fuerte que es el enrutamiento (routing), consistente en utilizar las URL para referirse a ficheros dentro del directorio raíz del servidor y después servir la vista necesaria al cliente. Aproximadamente 10.500.000 resultados en Google. (19)
- Drupal: Es un CMS modular con el que se pueden crear muchos tipos de páginas de forma sencilla. Se programa en PHP y se encuentra en su versión octava. Tiene más de 25.000 módulos disponibles en la página oficial y más de 750 temas para el diseño de nuestra web. Posee una de las comunidades más activas con 2500 actualización de código y 5000 comentarios de cuestiones de media a la semana. Aproximadamente 8.700.000 resultados en Google. (20)
- Ruby on rails: Tiene la particularidad de utilizar el lenguaje de programación Ruby. Utiliza el patrón MVC y su principal punto fuerte es la simplicidad de escribir código. Tuvo sus inicios en el año 2004 y actualmente se llega por la versión 4.0.3. Se basa sobre la premisa de no repetir código, y pudiéndose configurar de forma específica para cada proyecto. Existe un concepto que es “gems” que se traduce a un componente que facilitan la programación ágil de las partes específicas de tu propósito. Aproximadamente 6.760.000 resultados en Google. (21)
- Zend: Al igual que los Framework de PHP este también tiene la posibilidad de agregar componentes de manera fácil para así disminuir el tiempo de desarrollo. Empresas como Google o Microsoft han colaborado con este proyecto. Está en estos momentos por la versión 2. El número de módulos en este caso es claramente inferior, ya que la página oficial nos ofrece alrededor de 455 módulos. Su uso está en declive según ciertos informes. Aproximadamente 6.110.000 resultados en Google. (22)
- Yii: Tuvo su primera versión en 2008 y se llega por la 2 que está en Alpha. Nos permite añadir extensiones en forma de componentes o de widget y tiene integrado esquema de traducción de mensaje. Utiliza como patrones de diseño MVC y DAO (Objetos de Acceso a Datos) para extraer objetos de la base de datos que utilizemos. Podemos decir que es uno de los más potentes, además de tener una documentación excelente. Es de los más fácil de aprender de esta lista. En la página oficial existen más 1400 extensiones. Aproximadamente 5.690.000 resultados en Google. (23)
- Symfony2: Se construye con el patrón MVC y el lenguaje de programación PHP. En 2005 tuvo su primera versión estable y se llega por la versión 2.4.1. Permite la automatización de tareas gracias a la generación de código para la creación del modelo. Está diseñado más bien para proyectos grandes y de tipo comercial. En la página web oficial te puedes descargar junto con el Framework vienen todos los componentes ya listo para su uso. Aproximadamente 2.040.000 resultados en Google. (24)

Estos datos se han tomado en Marzo de 2014.

Existen muchas más como Mad Simple, Process wire, TYPO3, WordPress etc. que no se han sido objeto de estudio en este proyecto.

Tras un breve análisis de todas ellas, según el estudio realizado, vemos que existen en el mercado más aplicaciones con el lenguaje de programación PHP. (25; 26; 27) El Framework de Ruby on rails es muy atractivo por sus características para nuestro proyecto debido a que lleva desde el 2004 en el mercado, está en la versión 4.0.3 y es utilizado por muchos desarrolladores para sus proyectos. Yii, por su parte, empezaron con él hace 6 años, pero su repercusión y su crecimiento han sido de forma exponencial, sobre todo en este último año. Symfony y Drupal son buenas opciones, pero quizás estén más orientados a proyectos mucho más grande, no aportando nada al tipo de proyecto que estamos desarrollando.

Para concluir este análisis, indicar que elegimos el Framework de PHP Yii para nuestro proyecto. Esta elección se debe a su actual auge, además de que uno de sus puntos fuertes es la eficiencia, documentación y la gran cantidad de librerías de funcionalidades. El patrón de diseño que sigue es el de MVC, el cual nos permitirá la reutilización de código y separación de conceptos que nos ayuda a implementar de forma clara y sencilla. El propio Framework tiene una herramienta que nos facilita la creación de código automático para nuestros modelos, CRUD, formularios, controladores y módulos.

En el mundo del software, los proyectos no tienen una única solución. La tecnología es muy cambiante actualmente y surgen todos los días nuevas formas de hacer las cosas con lenguajes nuevos, frameworks mejorados etc. Es por ello que antes de empezar a picar código, debemos pensar dónde queremos llegar y cómo lo vamos hacer.

Primero hice un análisis de la tecnología actual, después encontré las funcionalidades de mi sistema en el análisis del usuario, y tras esto ya puedo decir que cuento con los dos ingredientes entre los que debo decidir cuál va ser la tecnología seleccionada. No siempre se puede elegir la tecnología óptima, por ejemplo, si esta tiene un coste muy elevado o bien en un lenguaje nuevo que debamos invertir un tiempo en aprenderlo, no se valorará como opción dentro del proceso de elección.

Para la programación de una aplicación web con estas características considero que debo emplear las siguientes herramientas:

1. Framework Yii: Con este framework podré utilizar todo su potencial sin tener que empezar de cero a crear una estructura de directorios, a crear entidades, validaciones, etc. Temas como la seguridad, al utilizar Yii, vienen con ciertos aspectos incorporados, como a veces estos aspectos son largo de implementar es trabajo que nos ahorramos, gracias a la comunidad que está detrás de este framework.
2. Mysql: Ya he utilizado este sistema de gestión de base de datos y me ha ido bien para proyectos pequeños. No he visto la necesidad de utilizar un sistema NO-SQL como Mongo DB u otro, ya que no tengo grandes volúmenes de datos, ni necesidad de velocidad de procesamiento en la base de datos.
3. Html5, css3, javascript: Html5 es un lenguaje básico que interpreta los navegadores. CSS3 nos proporciona las hojas de estilo que no permitirá embellecer la visualización. Y el último, Javascript, un lenguaje de programación que nos permite hacer página dinámicas y mejoras en la interfaz.

4. Foundation: Es un framework para el front-end. Se usará para hacer una página web responsive o también llamado adaptativo, quiere decir que se adapta a la pantalla de los dispositivos ya sea grande o pequeña. Gracias a esta tecnología no tengo que hacer una versión para cada tamaño de pantalla y esto hace que el desarrollo sea mucho más ágil. Además tiene otros elementos interesantes como formularios, navegación, estructuras de contenido, botones etc. los cuales voy a utilizar para este proyecto. (28)

Estas son las tecnologías con las que voy a trabajar a lo largo del proyecto. He cogido algunas que ya conocía como el lenguaje de programación Yii que se programa con php y MySQL server. En cambio, Foundation de Zurb es totalmente nuevo para mí y nunca he trabajado con herramientas que te permiten hacer web responsive. Los demás lenguajes como html, css y javascript ya los conocía con anterioridad.

4.2 Arquitectura y patrones

El modelo de arquitectura que voy a utilizar para la aplicación web es el modelo por capas muy utilizado hoy en día en el mundo del software. Se emplea en los sistemas para separar la lógica de negocio de la lógica de diseño. Anteriormente a este modelo se hacía todo en una capa, se hacían programas muy enrevesados, poco ágiles y muy difíciles de mantener. Con este modelo podemos hacer programas con buenos rendimientos, que tengan un mantenimiento fácil y que mejore sus funciones.

Esta arquitectura de cliente-servidor está segmentada en tres partes. Esta separación se hace para estructurar el programa y permite cambiar código en una de ellas sin que afecte a las demás. En un programa sin capas, la modificación en una parte del código obliga a revisar el resto del código. Utilizando este modelo por capas solo me preocuparé de forma localizada y me permitirá hacer cambios mucho más rápido. (29)

Para la elaboración de los siguientes puntos que voy a desarrollar, al igual que los apartados referidos a la aplicación web, he tomado como referencia tres proyectos. (30; 31; 32) Las tres capas que hemos hablado son las siguientes:

- Capa de presentación: También llamada capa web, capa de usuario, interfaz del usuario, es por así decirlo lo que ve el usuario, le transmite la información e interactúa con él. La capa de presentación se comunica solamente con la capa de negocio que definimos a continuación.
- Capa de negocio: también conocida como capa aplicativa, es donde residen lo que se ejecuta, se tienen las peticiones del usuario y se envían las respuestas tras el proceso. Es donde reside la lógica de negocio, donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestión de la base de datos.
- Capa de datos: también conocida como capa de base de datos, es donde acceden a los datos. Esta capa se comunica únicamente con la capa de negocio y es la única que accede directamente con los datos de la base de datos.

Las tres capas están relacionadas entre sí, como vemos en la figura 11, pero se comunican de forma específica.

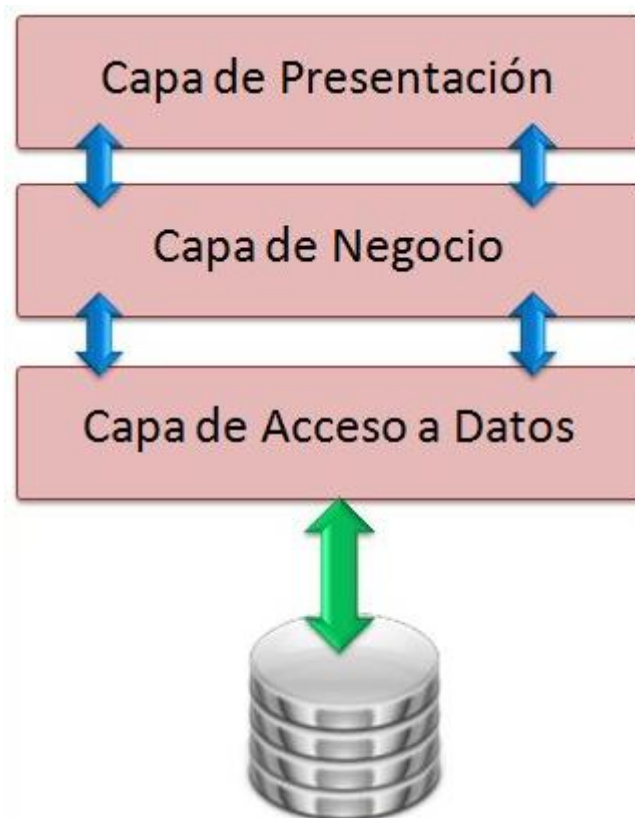


Figura 11 - Diagrama de la arquitectura de 3 capas. (33)

Estas capas se pueden implementar en varios servidores pero para este proyecto, inicialmente he decidido ponerlo en uno, esta arquitectura se llama por capas. Si fuese necesario, más adelante, analizo si es necesario, los beneficios e inconvenientes de separar las capas en varios servidores.

Como hemos visto, la capa que se interpone entre los datos y lo que ve el usuario es la capa de negocio, por lo que vamos a definir el diagrama de clases que lo compone, para hacer un diseño detallado de lo que tiene que hacer la aplicación.

Gracias al framework de Yii la capa de acceso a datos ya está incorporada por lo que no es necesario trabajar en ella.

El diagrama de clases define la estructura de un sistema, detallando las clases que les componen. Este diagrama se utiliza para proyecto con lenguajes de programación orientados a objetos. Una vez diseñado las clases con sus nombres y atributos debemos añadir los métodos que vamos a tener que implementar.

A continuación muestro las clases que voy a tener que implementar en la capa de negocio en la figura 12, 13, 14 ,15, 16 y 17.

Clase Modelo

Modelo
- nombre : String - modelo : int - imagen : String - descripcion : String - privado : Boolean - categoria : String - fechaCreacion : Date - pendiente : Boolean
+ insertar() : Void + editar() : Void + borrar() : Void + buscar() : Void + cambiarAlbum() : Void + donarModelo() : Void + esPendiente() : Void + descargar() : Void <u>+ findAll() : void</u> <u>+ save() : void</u> <u>+ update() : void</u> <u>+ <<destroy>> delete() : void</u> <u>+ findByPk() : void</u>

Figura 12 - Representación de la clase modelo en ARKivia web.

Esta clase representa los modelos que voy a guardar de los cliente. Con estos métodos implementos los casos de uso CU-5, CU-6, CU-7, CU-8, CU-9, CU-10 y CU-11.

Clase Álbum

Album
- nombre : String - imagen : String - descripcion : String
+ insertar() : Void + editar() : Void + borrar() : Void + descargar() : Void + descargarTodo() : Void <u>+ findAll() : void</u> <u>+ save() : void</u> <u>+ update() : void</u> <u>+ delete() : void</u> <u>+ findByPk() : void</u>

Figura 13 - Representación de la clase álbum en ARKivia web.

Con estos métodos la clase implemento los casos de uso CU-12, CU-13 y CU-14.

Clase Complemento

Complemento
- nombre : String - descripcion : String - complemento : String - modelo : int
+ insertar() : Void + editar() : Void + borrar() : Void + descargar() : Void <u>+ findAll() : void</u> <u>+ save() : void</u> <u>+ save() : void</u> <u>+ update() : void</u> <u>+ delete() : void</u> <u>+ findByPk() : void</u>

Figura 14 - Representación de la clase complemento en ARKivia web.

Esta clase implementa los casos de uso CU-15, CU-16 y CU-17.

Clase Peticion

Peticion
- emisor : int - receptor : int - mensaje : String - resolution : Boolean - pendiente : Boolean
+ pedirModelo() : Void + aceptarModelo() : Void + rechazarModelo() : Void <u>+ findAll() : void</u> <u>+ save() : void</u> <u>+ update() : void</u> <u>+ delete() : void</u> <u>+ findByPk() : void</u>

Figura 15 - Representación de la clase petición en ARKivia web.

Esta clase representa las peticiones que hacen los usuarios a los modelos. Los métodos implementan los casos de uso CU-18, CU-19 y CU-20.

Clase Usuarios

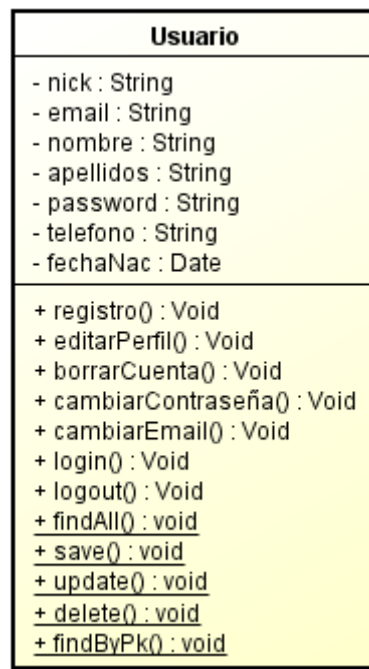


Figura 16 - Representación de la clase usuario en ARkivia web.

Esta clase hace referencia a los usuarios de ARkivia. Los métodos implementan los casos de uso CU-1, CU-4, CU-21, CU-22, CU-23, CU-24, CU-23.

En la figura 17 tenemos la representación de la clase log en el diagrama de clases.

Clase Log

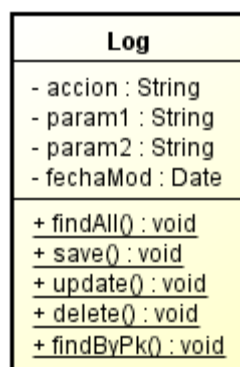


Figura 17 - Representación de la clase usuario en ARkivia web.

Esta clase sirve para guardar y visualizar la traza que van dejando los usuarios por las diferentes vistas.

Esta son las 5 clases principales que vamos a tener que implementar en nuestro proyecto, para poder interactuar entre la capa de vista y la capa de datos. No defino en los métodos los parámetros de salida, ni de entrada, dejando la tarea de asignarlos a la hora de programar. Faltan por definir las clases de Log, de Descarga, y otra para los caso de uso CU-2 y CU-3.

El caso de uso especial de Log solo tiene 5 campos, los atributos de id, idUsuario, acción, param1 y param2. El único método es el de insertar en la base de datos. La recogida de información para el análisis posterior se hará manualmente accediendo al gestor de base de datos.

A continuación vamos a explicar brevemente la estructura con la que trabaja Yii 1.1.13. En la figura 18 vamos a ver como la estructura general de las aplicaciones con este Framework.

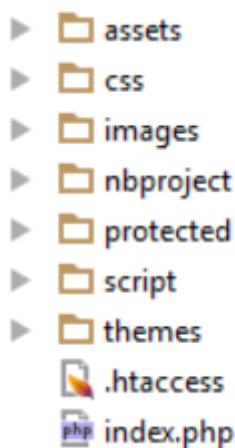


Figura 18 - Estructura de la carpeta principal de Yii.

La carpeta assets es la carpeta de recursos temporales como pueden ser javascript, hoja de estilos, imágenes y los modelos que visualizamos.

Las carpetas de css, images y script se alojan los archivos que utilizará la aplicación. En el html las direcciones para utilizar los recursos serán a estas carpetas.

La carpeta de protected es donde está la lógica del modelo y las vistas. Después la veremos con más detalle.

En la carpeta themes está la carpeta de foundation, que lo utilizo para mejorar el front-end y poder hacer una página responsive.

El archivo index.php es muy importante ya que es él que inicia Yii.

En la misma dirección donde se aloja el proyecto, debemos colocar la carpeta del framework de Yii. Esta carpeta contiene el núcleo del framework y es el que proporciona la lógica para así poder evitar reescribir todo el código de los procesos básicos que contiene. La idea de esto, es que solo implementamos los procesos específicos del sistema.

La siguiente carpeta es donde se guarda el núcleo de nuestra aplicación. Esta carpeta no es accesible, ya que contiene la lógica interna de nuestra aplicación. Dentro de la carpeta “protected” se encuentran las siguientes carpetas como vemos en la figura 19.

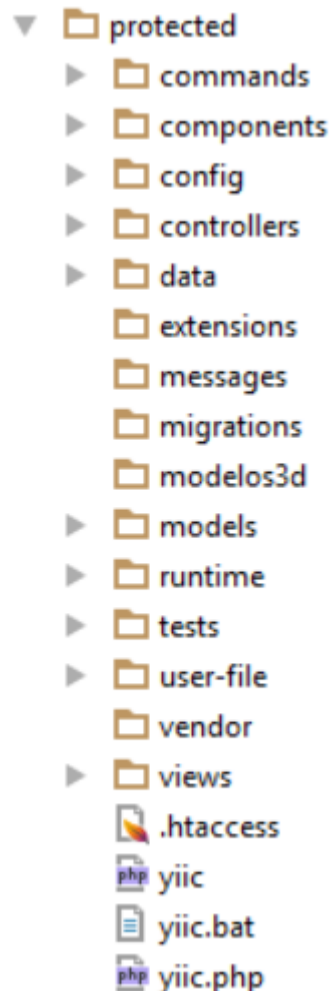


Figura 19 - Estructura de la carpeta “protected” de Yii.

En este proyecto solo se han necesitado las siguientes carpetas, no siendo todas ellas necesarias, vendor y extensiones, que son para añadir módulos, o la carpeta commands que nos sirve para ejecutar comandos en el servidor. No se han eliminado por sí en un futuro se necesitan.

La carpeta config está en los archivos de configuración main.php, donde se especifica el idioma estándar, el acceso a la base de datos, el acceso a la herramienta Zii y el formato de las urls entre otros parámetros.

A partir de ahora vamos a hablar exclusivamente de cómo Yii integra el MVC. Es por esto que seguimos el orden de primero la capa de modelo, a continuación el controlador y para finalizar la capa de vista. Lo hago de esta forma para poder ver el camino que llevan los datos del servidor hasta el cliente. Empezamos con el modelo, la carpeta “models” donde se alojan todas las clases que representan el sistema.

Estas clases se han generado automáticamente con Zii, una herramienta de Yii que permite su creación automática. Solamente indicando la tabla a la que hace referencia en la base de datos, genera la clase y todos sus atributos con las reglas que hayamos definido en la creación de la tabla. En algunos casos se ha modificado esta generación automática para ajustarlo a este proyecto.

En la siguiente imagen, la figura 20, vemos todos los modelos que se han creado.

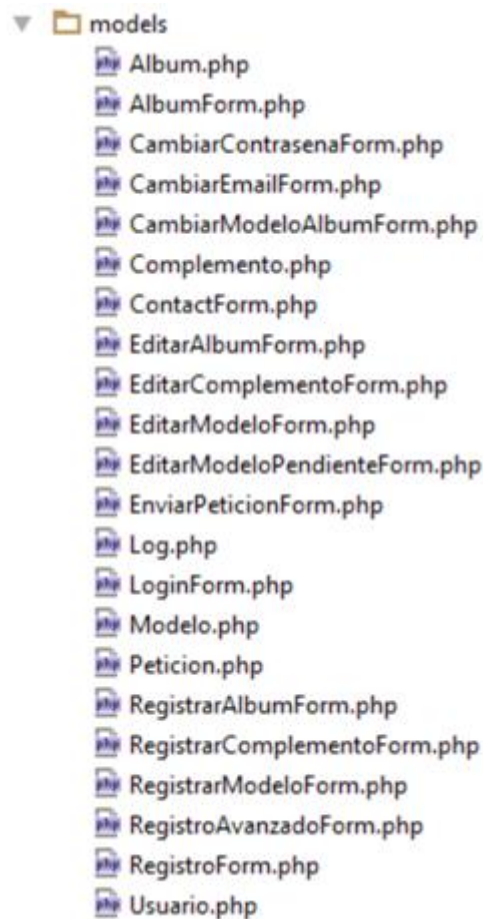


Figura 20 - Estructura de la carpeta “models” de Yii.

Como podemos ver en la imagen anterior, no solo se han creado las clases de los modelos, sino que también se han creado los archivos que contienen las clases de form que me permite hacer formularios a medida para cada caso específico. Esto lo he hecho así para poder manejar los formularios más fácilmente y poder personalizarlos.

Los de tipo form heredan de la clase CFormModel que vienen implementadas las funciones de rules, para las reglas de validación y attributeLabels para cuando se muestre el formulario las etiquetas no tengan el nombre de la columna de la base de datos.

Los modelos heredan de la clase CActiveRecord, esto me permite utilizar objetos de este modelo como representación de una fila en la base de datos. Muchas de las consultas de CRUD vienen dadas de las clases heredadas, como por ejemplo save, delete, findAll etc. Esto nos ahorra tiempo a la hora de programar y no tener que reinventar la rueda.

Tanto CFormModel como CActiveRecord heredan a su vez de CModel. Esta clase padre es de tipo abstracta ya que define que métodos y propiedades deben de tener las clases que herede de ella. En la figura 21 vemos su representación.

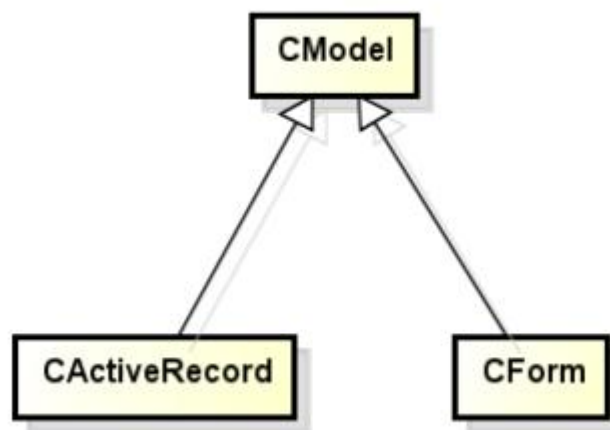


Figura 21 - Estructura de carpetas “models” de Yii. (31 pág. 37)

También he utilizado el patrón de diseño DAO (Data Access Object) que me proporciona Yii. La principal ventaja es que en la capa de negocios no es necesario saber cómo se estructura la información en la base de datos, trabajo con objetos. Otra de las ventajas de utilizar este patrón es si cambio a un nuevo sistema de gestor de base de datos. Este cambio no afecta al resto al código, solo cambiando el archivo de configuración y el acceso a la base de datos estará listo para funcionar con el nuevo gestor.

En la carpeta de controllers se encuentran las clases controller, con las que definiremos la lógica de la aplicación. En la figura 22 podemos ver las clases definidas.

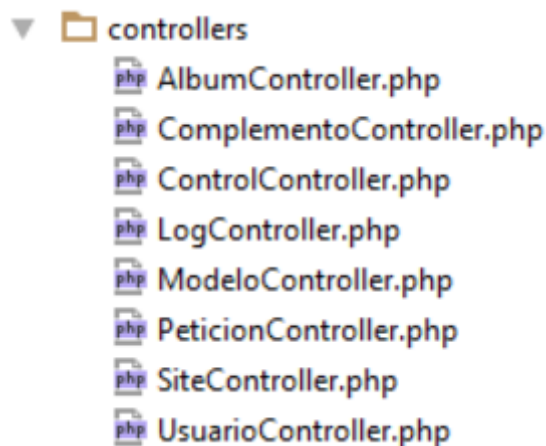


Figura 22 - Estructura de la carpeta “controllers” de Yii.

Como se puede ver, existe un controlador por cada modelo. He tenido que añadir el controlador de Site, que se usa para la lógica de las páginas antes del registro. Además de este, he tenido que añadir la clase ControlController de tipo controlador, siendo una clase sumamente como podemos comprobar más adelante en la parte de implementación.

Todas las clases del controlador heredan de la clase controller que es la que se encarga de poblar la tabla Logs. La siguiente imagen, la figura 23, muestra como están organizadas el diagrama de herencia de los controladores.

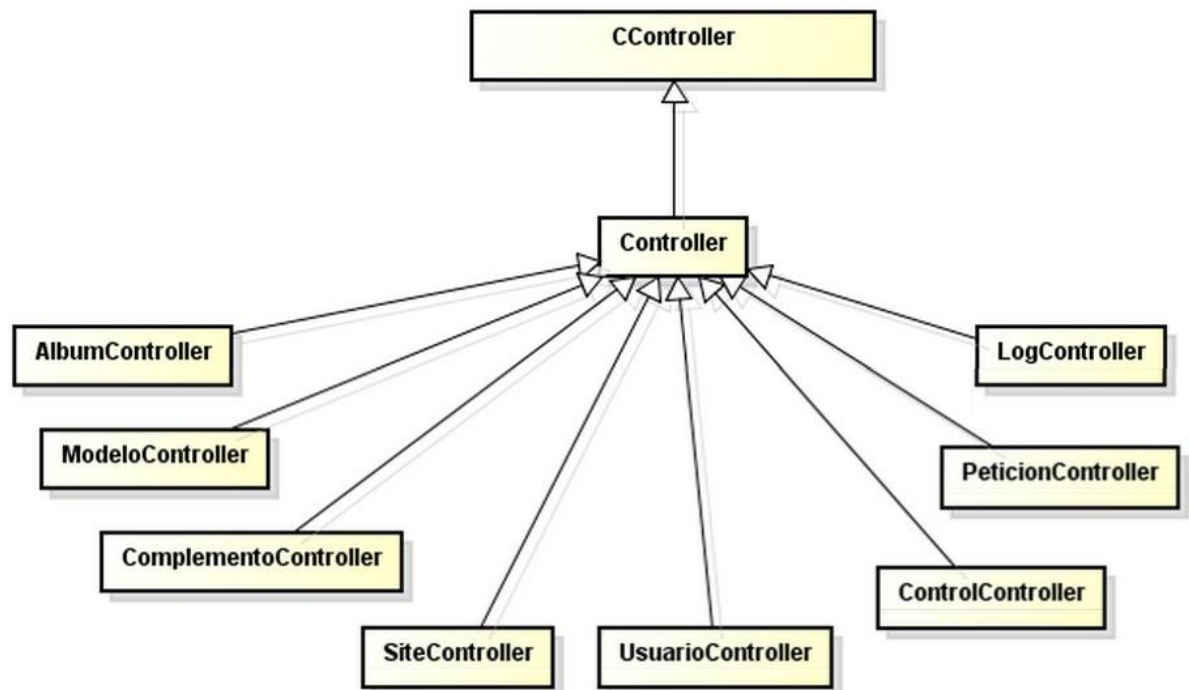


Figura 23 - Jerarquía de herencia de los controladores.

Esto lo hemos hecho así porque necesitamos que cada acción quede registrada en la tabla de log, de tal modo que cada vez que se invoca una acción del controlador guarde qué acción y los parámetros que se han enviado. La clase **controller** hereda de la clase **CController** de Yii que es la que se encarga de gestionar los controladores y dotarles de sus funcionalidades.

Para finalizar, describo la capa de vista, que está asociada a la carpeta de **views**. En la siguiente imagen, la figura 24, vemos la estructura que tiene.

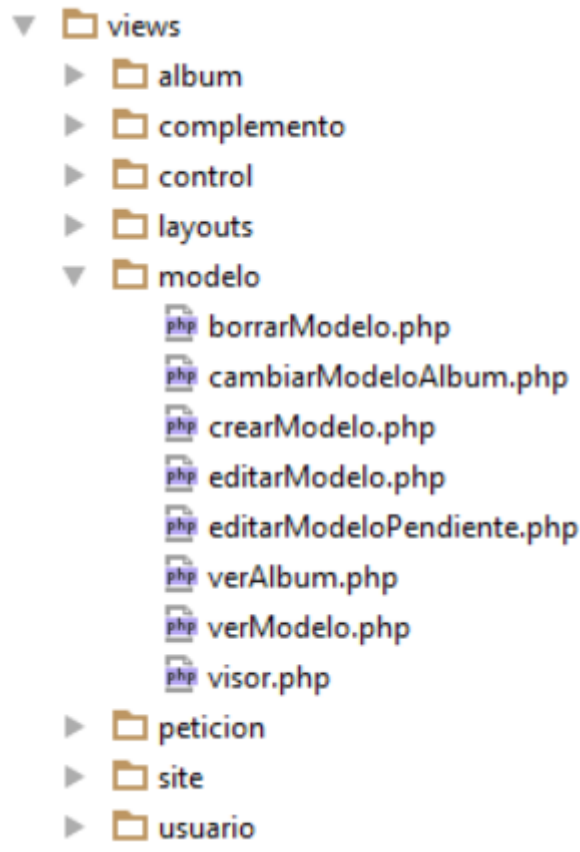


Figura 24 - Estructura de la carpeta “views” de Yii.

Como se puede apreciar dentro de la carpeta “modelo”, los archivos estan nombrados por sus funcionalidades. También se han añadido varias carpetas como la de site, control y layouts, para completar todas las vistas que tiene la aplicación web.

Para mostrar una vista desde el controlador utilizamos el método render, de la clase heredada CController. Los parámetros de render son el nombre de la vista y un array de clave-valor donde la clave es el nombre de la variable que vamos a utilizar en la vista y el valor, el objeto que se va a utilizar en la vista.

Llegado a este punto, hemos hablado de las 3 capas y cómo se organizan en Yii. Tenemos definido nuestro diagrama de clases y nos queda hablar de la base de datos y de la navegación de la aplicación.

4.3 Base de datos

Este apartado trata del diseño que hemos tenido que realizar para que el sistema funcione, tal y como lo requiere. Sin un previo análisis es imposible hacer una base de datos correcta, ya que debemos modelar la realidad en forma de modelo de dominio, y una vez tengamos este, debemos materializarlo en la base de datos. Una vez creada la base de datos debe de ser capaz de absorber toda la información que el cliente desea almacenar. Si no es capaz de implementar un caso de uso, tal vez sea por un mal diseño de la base de datos.

El modelo de dominio que tenemos en el apartado 3.5 de este documento muestra las relaciones entre los elementos que tiene el sistema. A mayores de esto, debemos agregar una nueva funcionalidad pedida como requisito del tutor, que es la de guardar un registro de lo que se hace en la aplicación web para poder dar estadísticas, saber los últimos modelos vistos, los modelos más descargados. Es por ello que agregamos una tabla a mayores llamada log, que contendrá todos los registros que se desea almacenar de la aplicación.

El esquema de la base de datos se puede ver en el anexo de diagramas la ilustración 5.

Como podemos ver se mantienen las relaciones del modelo de dominio y se añade la relación que existen en la tabla log, con la tabla de usuarios a través del campo idUsuario.

Gracias a la facilidad que nos ofrece el framework de Yii, si surge un problema, ya sea en la implementación o en producción, los cambios de base de datos y de modelo se realizan de forma rápida sin afectar al resto de elemento. Esta ventaja es clave, si lo que pretendemos es lograr una rápida adaptación a nuestro entorno.

4.4 Navegación en la aplicación

La aplicación web debe de tener un esquema de navegación bien definido y de fácil uso. Es muy importante que los usuarios puedan intuir cómo realizar las acciones, para poder ir más rápidos y estar en una página agradable.

Voy a utilizar un enfoque descendente, es decir voy a partir de los más general es de la página, como pueden ser la vista principal y el panel de control y después ir bajando al detalle donde están los formularios donde se realiza las acciones más específicas.

Empezamos con el área del usuario no registrado. El esquema se muestra en la figura 25 .

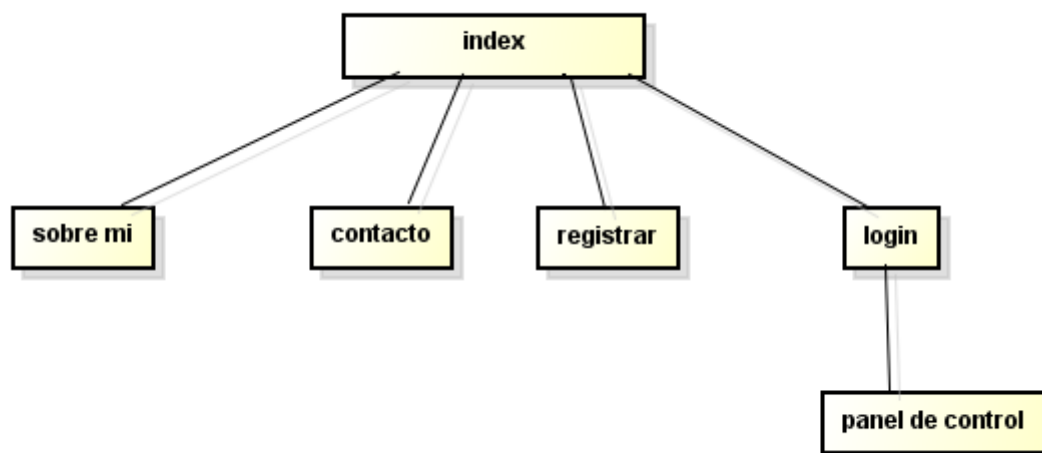


Figura 25 - Workflow de la área de usuario no registrado.

Como podemos ver la página principal sería el index, desde ahí se puede llegar a las acciones "sobre mi", "contacto", "registro" y "login". Si realizamos el login correctamente podemos acceder al panel de control, que está dentro del área del usuario.

El siguiente esquema sería para los usuarios que ya se han logueado y pueden disfrutar de todas las funcionalidades de ARkivia. El panel de control ofrece acceso rápido a todas gestiones de álbum, perfil, modelo, galería, contacto y peticiones. En la figura 26 podemos apreciar la navegación desde el panel de control.

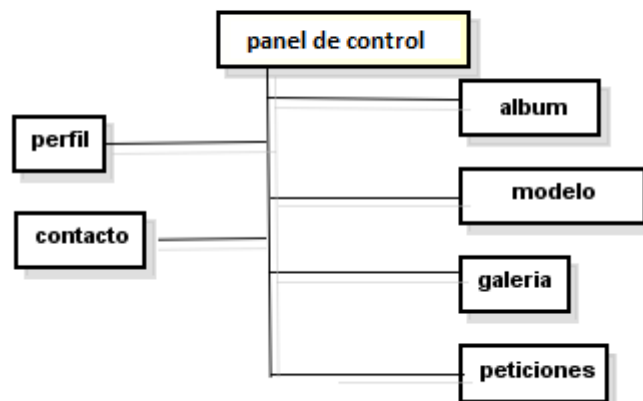


Figura 26 - Workflow de la sección desde el panel de control.

Al acceder al panel de control es interesante poder ir a los elementos generales de forma rápida, por eso ponemos todas las posibilidades sin entrar en detalle. Aunque luego se pongan accesos directos a los álbumes o modelos más utilizados, es necesario poner los grandes grupos para que los usuarios tengan claro adónde van.

Ahora procedo a ir a cada uno de ellos para ver un nivel más de profundidad. Empezamos con álbum en la figura 27

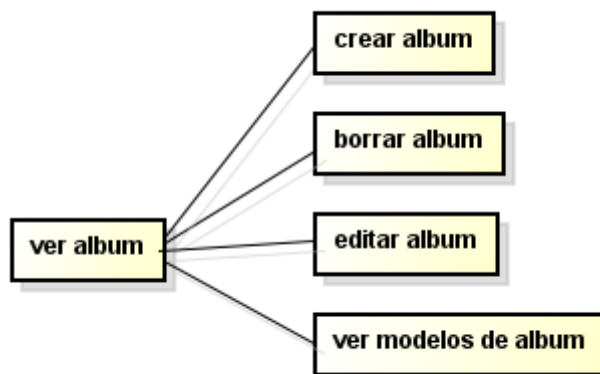


Figura 27 - Workflow de la sección de álbum.

Después defino el diagrama los modelos en la figura 28:

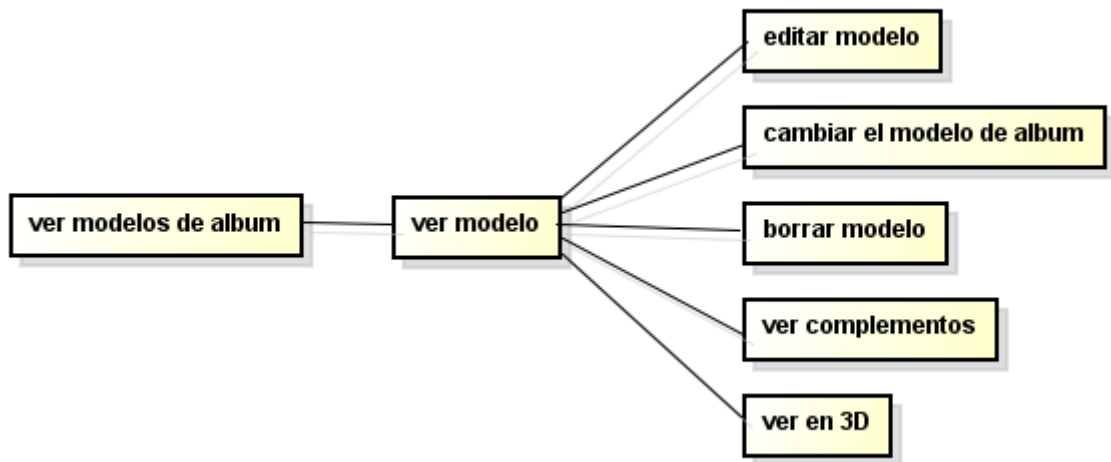


Figura 28 - Workflow de la sección de modelo.

A continuación el diagrama de complementos en la figura 29.



Figura 29 - Workflow de la sección complemento.

El diagrama de navegación de galería está en la figura 30



Figura 30 - Workflow de la sección galería.

El perfil de usuario tendrá una navegación con el esquema de la figura 31

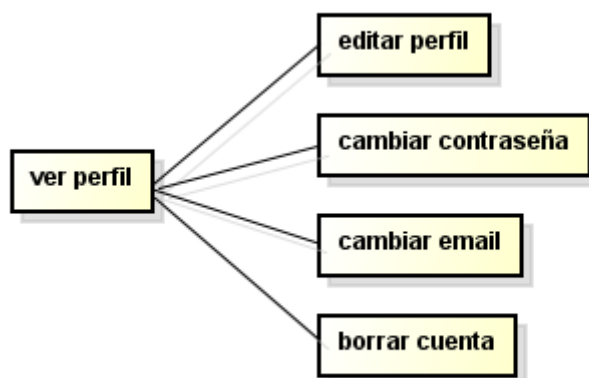


Figura 31 - Workflow de la sección de ver perfil.

Una vez he tomado en cuenta todo el esquema de navegación de navegación, procedo a la creación de la barra de navegación que va a estar disponible, siempre, en la parte superior del navegador. Esta barra debe permitir llegar a cualquier parte de forma rápida e intuitiva. La figura 32 muestra las diferentes opciones.

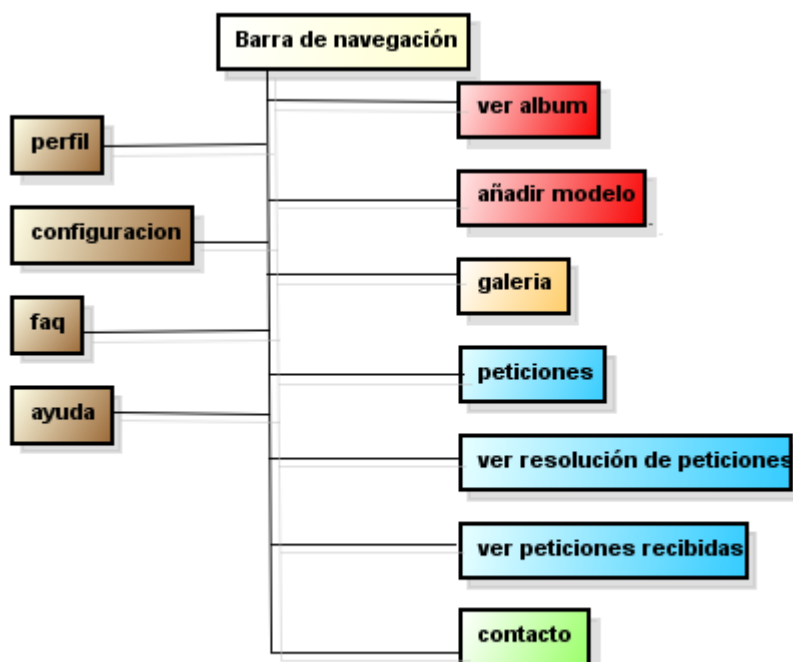


Figura 32 - Workflow de la barra de menú superior.

Una vez realizado estos diagramas debemos contrastarlo con el comportamiento de los usuarios reales. Una técnica muy utilizada es utilizar Google Analytic para estudiar los comportamientos que tienen los usuarios con la aplicación. Si vemos que los usuarios acceden más a una página, que a otras, debemos estudiar el poner un acceso directo o bien añadirla a la barra de navegación. Esto no se podrá llevar a cabo en este proyecto, pero si en el futuro cuando esté en producción.

4.5 Diagramas de secuencias y diagramas de estado

En este apartado vamos a realizar el diseño de los diagramas de secuencia, que son un modelaje de las interacciones que existen en el sistema entre los modelos. Debido a su gran tamaño, hemos decidido unos ejemplos en el anexo en el apartado de diagramas. Las ilustraciones 2, 3 y 4 representan los casos de uso de ver el perfil, cambiar de contraseña y eliminar la cuenta. El resto de diagrama de secuencia se encuentra en el contenido del CD-ROM, en la carpeta de desarrollo, en el archivo .asta.

Otro de los diagramas que vamos a diseñar, son los llamados diagramas de estado. Estos diagramas representan los diferentes estados que van pasando un objeto a lo largo de su vida útil. Los cambios de estados son producidos por la representación de la ejecución de un método. En nuestro caso tenemos dos maquinas de estado.

El primer caso, hace referencia al usuario y cómo va cambiado de estado, según va realizando ciertas acciones como la de registrarse, activarse, desactivar la ayuda, cambiar de contraseña o de email y/o eliminar su cuenta. El diagrama de estado del usuario lo podemos ver en el anexo en la parte de diagramas la ilustración 6.

El segundo caso es el caso de una petición de un modelo. Vamos a ver cómo va pasando de estado y viendo los casos de petición aceptada y petición rechazada. La figura 33 muestra el diagrama de estado de petición de un modelo:

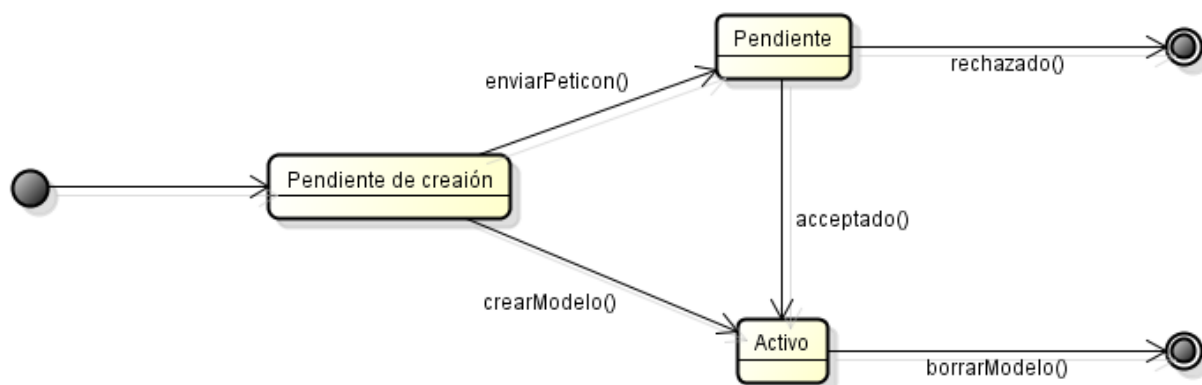


Figura 33 - Diagrama de estado de una petición en ARkivia web.

5. Implementación

5.1 Proceso de implementación

Una vez realizado el diseño, debo crear el código para que la aplicación funcione tal y como desean los clientes. Es la parte donde vamos a construir la aplicación con los diseños que hemos hecho hasta ahora. Si lo he hecho correctamente, no debería haber dudas sobre el qué debo programar.

En este software, algunas herramientas que he utilizado, son nuevas para mí y he tenido que invertir tiempo en aprender cómo funcionan y cuál es la forma correcta de trabajar con ellas.

Primero me he enfrentado al framework de Yii, donde he utilizado la documentación oficial de su página. También he documentado con otras fuentes, o viendo posibles preguntas de otros usuarios del framework. Todo ello me ha permitido aprender, en dos semanas, lo básico de Yii y cómo funciona la conexión con la base de datos, la creación de entidades, utilizar la carpeta assets, creación y validación de formularios de forma automática y entender la estructura interna.

Una vez obtenidos los conocimientos básicos de Yii, he implementado el sistema separándolo en las siguientes tareas:

- Creación de base de datos.
- Creación del modelo.
- Creación de la vistas.

Para la creación de la aplicación web no he seguido este orden sino que he seguido la secuencia de casos de uso. El proceso que he seguido exactamente lo explico en el siguiente párrafo.

Primero he construido la base de datos, y después he creado las entidades. Una vez realizados estos dos pasos, he ido eligiendo un caso de uso, después creaba el acceso a la vista y a continuación ponía los elementos básicos de la vista para que el caso de uso funcione. Una vez realizado el paso anterior, hacia el modelo y su acceso a la capa de datos. Probaba el caso de uso, y corregía errores hasta que funcionaba, como pedía la definición del caso de uso. Para finalizar, he embellecido la vista y probado su integración con los demás casos de uso. Este proceso lo he repetido para cada caso de uso definido.

5.2 Implementación de casos específicos

En este apartado, hablaremos sobre cómo actuar aplicando la solución antes diferentes casos especiales. Daremos 2 ejemplos, el primero es sobre la estructura en la que se guardan los modelos de los usuarios, y la segunda, sobre la implementación del servicio web para poder transmitir información a la aplicación móvil.

El primer caso hace referencia a la carpeta “User-file” dónde está el contenido de todo los archivos guardados por los usuarios. En la figura 34 podemos ver un ejemplo.

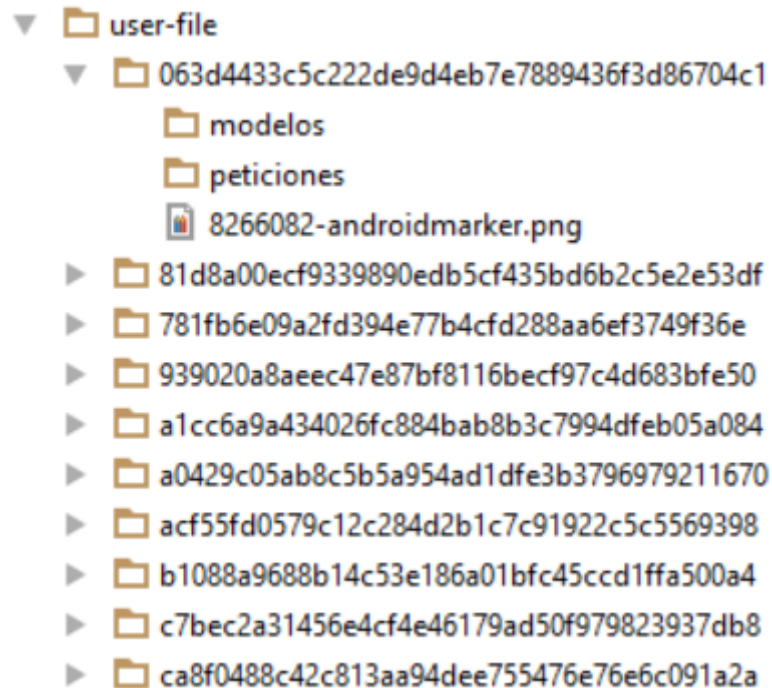


Figura 34 - Estructura del directorio privado de los usuarios, modelos, complementos y peticiones.

Hemos organizado la carpeta por un id de usuario, que en este caso es el e-mail, y los algoritmos de SHA-1 y RSA. Cada usuario tiene su carpeta y en su interior se alojan sus modelos con sus complementos. Además es necesaria la carpeta de peticiones, ya que si el usuario hace una petición de un modelo tiene que hacer una copia en ese instante y guardarla en su directorio. Imaginemos que el usuario que tiene el modelo le hacen una petición de un modelo y este le cambia. Si el usuario acepta la petición debe de ser del modelo que antiguo, y no del nuevo. Esta es la razón por la que se guarda en la carpeta del que envía la petición.

Ahora hablamos del segundo caso, donde en el apartado de diseño hablamos de del controlador “*controlControler*” y no desarrollamos el por qué de su existencia. Ahora explicamos las dos funciones que tiene:

La primera es la que implementa la descarga de los álbumes, modelos y complementos de los usuarios. Dependiendo de la consulta que se haga, la lógica es diferente para cada uno de los casos.

La segunda función es la de hacer la conexión con la aplicación móvil. Utilizamos el protocolo cliente/servidor sin estado REST. El envío se produce a través de una petición, con parámetros específicos si fuesen necesario. El servidor procesa la información y envía un mensaje con formato JSON respondiendo al envío.

Son tres las funciones que se implementan en este caso.

- `getAppListAlbum`: Devuelve la organización del álbum y de los modelos que tiene el usuario. Los parámetros que son enviados, son el id del usuario.

- `getAppModelo`: Si existe devuelve el contenido del archivo del modelo y del archivo de apoyo `.mtl`. Los parámetros que se envían son el id del usuario y del modelo. Esto sirve para comprobar que el modelo es de este usuario.
- `getAppUsu`: Devuelve el id del usuario si existe. Uno de los parámetros es la contraseña, que se debe aplicar una clave privada única que solo existe en la aplicación web y en la aplicación móvil. El otro parámetro que se envía es el email.

Veremos su uso con más detalle en el apartado de la aplicación móvil.

5.3 Otras herramientas utilizadas

En cualquier proyecto es recomendable utilizar cosas ya existentes para no tener que reinventar la rueda para cada funcionalidad. Por eso utilizo varias frameworks o herramientas para implementarlo en mi proyecto y no tener que hacerlo de nuevo. En esta apartado hablaré sobre el framework de Foundation y el visor de objetos 3D JSC3D.

5.3.1 Foundation

Es un framework de front-end que tiene tres ventajas. La primera es que es responsive, y permite que programando una única aplicación se implemente para todos los tipos de dispositivos como pueden ser smartphone, tablets y diferentes monitores. La segunda ventaja es que se programa muy rápido, ya que la sintaxis es muy sencilla e intuitiva.

Los elementos que he utilizado de Foundation son los botones, las tablas, la paginación, la barra de menú, los bloques de mensajes, los layout para que sea web responsive y elementos de formularios.

La implementación de este framework es muy sencilla ya que solo debemos introducir los archivos javascript que utilizamos en cada página para que estos funcionen. Se basa en la librería JQuery para su funcionamiento, por lo que debemos ingresarla también en las librerías de javascript importadas.

5.3.2 JSC3D

Es un proyecto que utilizo en mi aplicación web para visualizar los modelos 3D de los clientes. Utiliza javascript y HTML5 canvas como lienzo, donde se representan los modelos. Es compatible con varios navegadores como Firefox, Safari, IE9 y Chrome, entre otros. Esta aplicación es muy sencilla, configuras los parámetros en javascript e introduces la dirección del modelo que quieres ver, no tienes que hacer nada más. (34)

5.3.3 PclZip

Para poder comprimir todos los archivos en caso de que el usuario quiera guardar sus modelos de ARKivia, utilizamos PclZip. Esta librería tiene un funcionamiento muy simple, que consiste en crear primero el archivo de compresión dándole un nombre, y después puedes ir añadiéndole direcciones de archivos y carpetas. En este caso hago especial hincapié en no dejar residuos en el servidor, ya que se crean carpetas temporales que debo borrar una vez que la descarga se realice.

5.3.4 BreadCrumb

Esta librería me permite tener en todas las vistas unos hipervínculos, que me permiten ir a acciones anteriores. Lo que hace es guardar la url y un nombre por cada petición que se hace. Algunas de las peticiones resetean el contenido y vuelven a empezar desde la acción principal home. Tiene una carpeta que contiene la vista del breadCrumb para poder modificarlo. Se pueden configurar qué páginas son las que resetan el BradCrumb.

Durante la implementación de ARKivia web se utilizó el siguiente software:

- NetBeans 7.3: Entorno de desarrollo para PHP.
- XAMMP: De la cual se ha utilizado como servidor Apache, la base de datos de MySQL y el servidor de correo Mercury.
- GIMP: Editor de imágenes.

6 Pruebas de validación

La batería de pruebas que hemos realizado para la aplicación web es muy importante, ya que me asegurará la funcionalidad de los requisitos del sistema. Los ensayos que se hacen, no solo son para verificar el funcionamiento, sino que también tengo en cuenta el tiempo que utiliza en realizarlo.

La finalidad de este apartado es comprobar que los resultados dados por la aplicación son los esperados, y en caso de que no sea así, corregir los errores. En caso de detectar un error de requisitos mínimos, debo solucionarlo primero evaluando su gravedad y su coste en esfuerzo para arreglarlo. Al finalizar la evaluación aplicó una solución.

Este plan de pruebas se debe aplicar, parcial o totalmente, cuantas veces sea necesaria hasta conseguir que la aplicación cumpla con los requisitos impuestos por el usuario. Se pueden diferenciar 2 tipos caso de error, el que afecta únicamente a sí mismo, se dice que es de carácter independiente, o que el error se propague a las demás pruebas, y en este caso es de carácter dependiente. Estas segundas suelen ser más difíciles de arreglar ya que debemos comprobar la relación entre las pruebas, por lo que generalmente se hace más larga la corrección.

Este proceso de pruebas de validación es cíclico y variable, ya que el número de pruebas puede aumentar debido al añadido de nuevas funcionalidades. Este proceso puede requerir mucho tiempo, por lo que nos centramos en la primera aplicación de las pruebas de validación.

Existen 2 tipos de pruebas, las de caja negra y las de caja blanca:

- Caja negra: Enfoque funcional. Demuestran que las funciones de la aplicación son operativas, que los datos de entrada son válidos y la salida es la correcta.
- Caja blanca: Enfoque en la lógica del sistema. Se han de proponer casos en los que se ejecutan conjuntos específicos de condiciones.

En las pruebas de caja blanca, los datos introducidos deben de ser reales e intentar encontrar valores que hagan fallar a la aplicación. Existen datos llamados fronteras, que son los que normalmente se prueban, y son los más cercanos a producir un error. Por ejemplo, si nos permite números enteros del 0 a 100 tenemos que probar con el 101 y con el -1.

Las pruebas han de ser diseñadas de forma secuencial debido al sentido de la aplicación. Por ejemplo, no se puede comprobar una registro de complementos de un modelo, si no existen modelos en la aplicación.

6.1 Metodología

En este caso, me centro en los aspectos funcionales de la aplicación comprobando cada uno de los requisitos mínimos y su funcionalidad. Por cada requisito funcional definido anteriormente en el apartado 3.6 de este documento haremos unas pruebas para determinar el correcto funcionamiento de este.

Por cada prueba definimos los siguientes aspectos de ella:

- Nombre
- Requisito funcional relacionado
- Datos de entrada
- Datos de salida esperados
- Datos de salida obtenidos
- Tiempo de procesamiento.
- Comentario

La aplicación de la batería de pruebas se han realizado el siguiente entorno:

- Sistema operativo: Windows 8.1.
- Navegador web: Mozilla Firefox 37.0.1.
- Conexión a Internet: 6 MB subida / 1 MB de bajada

6.2 Validación de la aplicación

El índice de la batería de prueba es el siguiente:

1. Área de visitantes
 - 1.1. Registro de usuarios
 - 1.2. Validación del email
 - 1.3. Login
 - 1.4. Formulario de contacto
2. Área de usuarios
 - 2.1. Modelos
 - 2.1.1. Insertar modelo
 - 2.1.2. Editar modelo
 - 2.1.3. Suprimir modelo
 - 2.1.4. Descargar modelo
 - 2.1.5. Donar modelo
 - 2.1.6. Cambiar modelo de álbum
 - 2.1.7. Buscar modelo
 - 2.1.8. Añadir modelo desde galería
 - 2.2. Álbum
 - 2.2.1. Insertar álbum
 - 2.2.2. Editar álbum
 - 2.2.3. Suprimir álbum
 - 2.2.4. Descargar un álbum
 - 2.2.5. Descargar todos los álbumes
 - 2.3. Complemento
 - 2.3.1. Insertar complemento
 - 2.3.2. Editar complemento
 - 2.3.3. Suprimir complemento

- 2.3.4. Descargar complemento
- 2.4. Petición de modelo
 - 2.4.1. Pedir un modelo
 - 2.4.2. Pedir un modelo rellenando los datos previamente
 - 2.4.3. Rechazar una petición
 - 2.4.4. Aceptar una petición
- 2.5. Perfil
 - 2.5.1. Editar perfil
 - 2.5.2. Cambiar email
 - 2.5.3. Cambiar contraseña
 - 2.5.4. Borrar perfil donando los modelos
 - 2.5.5. Borrar perfil sin donar los modelos
 - 2.5.6. Borrar perfil donando los modelos públicos
- 2.6. Logout
- 3. API
 - 3.1. Comprobar usuario y contraseña
 - 3.2. Devolver jerarquía de álbum.
 - 3.3. Devolver modelo.

En total son 35 pruebas las que debemos contemplar para hacer validar el prototipo creado:

Referencia	P-1.1
Nombre	Registro de usuario
Requisito mínimo relacionado	RF-23
Datos de entrada	<ul style="list-style-type: none"> • Nick: "Sandra22" • Email: sandra@gmail.com • Password: Paris21 • Repet password: Paris21
Datos de salida esperados	Se envía un mensaje al correo para validarlo. Tiene que insertarse un registro en la base de datos con los datos introducidos.
Datos de salida obtenidos	No se envía ningún mensaje. Introduce correctamente los datos del usuario en la base de datos.
Tiempo de procesamiento	Irrelevante.
Comentario	<p>El servidor web no estaba instalado en el servidor en producción por lo que se va a llevar a cabo una modificación de este proceso y se va a validar el usuario en la página siguiente del registro dando a un link de activación.</p> <p>Se guardará el código que funciona en local para integrarlo en la próxima versión del prototipo.</p>

Referencia	P-1.2
Nombre	Validar el usuario
Requisito mínimo relacionado	RF-23
Datos de entrada	<ul style="list-style-type: none"> Clic en el link de activar.
Datos de salida esperados	Se actualiza el registro y pasa a ser un usuario activado.
Datos de salida obtenidos	Se realiza correctamente ya que el registro se modifica a estado de activado.
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-1.3
Nombre	Login
Requisito mínimo relacionado	RF-24
Datos de entrada	<ul style="list-style-type: none"> Email: sandra@gmail.com Password: Paris21
Datos de salida esperados	Se abre una sesión con el usuario y nos permite entrar en nuestro panel de control principal.
Datos de salida obtenidos	Crea una nueva sesión y nos redirecciona al panel de control de ARKivia.
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-1.4
Nombre	Formulario de contacto
Requisito mínimo relacionado	Añadido posteriormente
Datos de entrada	<ul style="list-style-type: none"> Nombre: Sandra Email: sandra@gmail.com Asuntos: Problema con el registro Texto: Me sale un error 404. ¿Qué puedo hacer?
Datos de salida esperados	Se envía un correo con los datos introducidos a nuestro email.
Datos de salida obtenidos	No se envía ningún email
Tiempo de procesamiento	Irrelevante.
Comentario	Debido al servidor de correo del servidor en producción no se puede enviar email. En el próximo prototipo se intentara solucionar este problema.

Referencia	P-2.1.1
Nombre	Insertar modelo
Requisito mínimo relacionado	RF-1
Datos de entrada	<ul style="list-style-type: none"> • Elegir álbum: Nuevo • Nombre: Dragón • Archivo obj: tamaño - 462KB • Descripción: Dragón enorme • Privado: Activado • Imagen: tamaño - 4KB
Datos de salida esperados	Crea un registro con los datos introducidos y nos muestra la página con un link para ver el modelo con el mensaje de que se ha introducido correctamente.
Datos de salida obtenidos	Realiza la acción según lo esperado
Tiempo de procesamiento	10 segundos
Comentario	El tiempo es razonable en este caso. Se ha probado introducir modelos más pesados para probar su velocidad. Se ha encontrado una relación que de cada MB se tarda unos 20 segundo hasta un máximo de 2 megas. Si fuese necesario se aumentaría en el archivo php.ini este tamaño máximo.

Referencia	P-2.1.2
Nombre	Editar modelo
Requisito mínimo relacionado	RF-2
Datos de entrada	<ul style="list-style-type: none"> • Elegir álbum: Proyecto ToolYGame • Nombre: Interrogante • Archivo obj: tamaño - 129KB • Descripción: Un interrogante • Privado: Desactivado
Datos de salida esperados	Se cambia el registro de la base de datos y además se borra el anterior modelo para sustituirle por el nuevo. Se redirecciona a la vista del modelo modificado
Datos de salida obtenidos	Hace correctamente lo esperado
Tiempo de procesamiento	5 segundos
Comentario	El tiempo de procesamiento es correcto ya que tarda menos que el modelo del dragón al ser menos pesado.

Referencia	P-2.1.3
Nombre	Suprimir modelo
Requisito mínimo relacionado	RF-3
Datos de entrada	<ul style="list-style-type: none"> Clic en el link de suprimir modelo
Datos de salida esperados	En caso de que tenga complemento nos pregunta si queremos borrar también los complementos. En caso contrario de que no tenga complementos se borra el registro del modelo y borra del servidor el modelo.
Datos de salida obtenidos	Realiza la operación según lo esperado.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.1.4
Nombre	Descargar comprimir modelo
Requisito mínimo relacionado	RF-4
Datos de entrada	<ul style="list-style-type: none"> Clic en descargar modelo
Datos de salida esperados	Nos permite guardar el modelo en formato comprimido con todos los complemento si los tiene.
Datos de salida obtenidos	Sale una ventana del sistema operativo para guardar el modelo en formato .rar. Dentro se encuentra el modelo con todos los complementos.
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.1.5
Nombre	Donar modelo
Requisito mínimo relacionado	RF-5
Datos de entrada	<ul style="list-style-type: none"> Donar modelo
Datos de salida esperados	Pasa el modelo a pertenecer al usuario ARkivia y de compartirlo con la comunidad de ARkivia. Actualiza la página mostrando un mensaje dando las gracias por la donación
Datos de salida obtenidos	El sistema actúa correctamente.
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.1.6
Nombre	Cambiar modelo de álbum
Requisito mínimo relacionado	RF-6
Datos de entrada	<ul style="list-style-type: none"> • Seleccionar un álbum existente
Datos de salida esperados	Pasa el modelo de estar en un álbum a estar en el álbum seleccionado. Al realizarse la acción pasa a la vista del modelo.
Datos de salida obtenidos	La salida es la esperada
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.1.7
Nombre	Buscar modelo
Requisito mínimo relacionado	RF-7
Datos de entrada	<ul style="list-style-type: none"> • Buscar: "dragón"
Datos de salida esperados	Busca en la base de datos de ARkivia todos los modelos públicos que contengan la palabra "dragón" en el título o en la categoría. Muestra el resultado en la misma pantalla.
Datos de salida obtenidos	El sistema actúa de forma correcta.
Tiempo de procesamiento	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.1.8
Nombre	Añadir modelo desde galería
Requisito mínimo relacionado	RF-8
Datos de entrada	<ul style="list-style-type: none"> • Clic en el link
Datos de salida esperados	Pasa el modelo de estar en un álbum a estar en el álbum seleccionado. Al realizarse la acción pasa a la vista del modelo.
Datos de salida obtenidos	Error - No encuentra el album.
Tiempo de procesamiento	Irrelevante.
Comentario	El error es causado porque no existe el albúm con nombre sin clasificar. En este caso no es el módulo que falla sino que al crear el usuario con el que se ha hecho las pruebas no se creaba la carpeta sin clasificar y esto hace que esta prueba de error.

Referencia	P-2.2.1
Nombre	Insertar album
Requisito mínimo relacionado	RF-9
Datos de entrada	<ul style="list-style-type: none"> Nombre: Proyecto X Descripción: Proyecto para la empresa imagen: tamaño - 3KB
Datos de salida esperados	Crea un registro de álbum con los datos introducidos y redirección a la página de álbumes.
Datos de salida obtenidos	Realiza la acción según lo esperado
Tiempo de procesamiento.	1 segundos
Comentario	El tiempo es despreciable siempre y cuando el peso de la imagen no pasa de 1 MB.

Referencia	P-2.2.2
Nombre	Editar álbum
Requisito mínimo relacionado	RF-10
Datos de entrada	<ul style="list-style-type: none"> Nombre: Proyecto luz Descripción: Proyecto para la empresa de luz Imagen: tamaño - 23KB
Datos de salida esperados	Se cambia el registro de la base de datos y además se borra la anterior imagen para sustituirla por la nuevo. Se redirección a la vista del álbum modificado
Datos de salida obtenidos	Hace correctamente lo esperado.
Tiempo de procesamiento.	1 segundos
Comentario	Mismo caso que el anterior.

Referencia	P-2.2.3
Nombre	Suprimir álbum
Requisito mínimo relacionado	RF-11
Datos de entrada	<ul style="list-style-type: none"> Clic en el link de suprimir álbum
Datos de salida esperados	Al hacer clic se debe preguntar al usuario si quiere trasladar sus modelos a otro álbum. Si es que no, procede a borrar todos los elementos con sus complemento incluido el álbum. Si es que si debe trasladar los modelos a la carpeta "Sin clasificar"
Datos de salida obtenidos	Realiza la operación según lo esperado.
Tiempo de procesamiento.	Irrelevante.
Comentario	Si el álbum que queremos borrar es "Sin clasificar" el sistema debe mostrar un mensaje de error. Se ha comprobado.

Referencia	P-2.2.4
Nombre	Descargar un álbum
Requisito mínimo relacionado	RF-12
Datos de entrada	<ul style="list-style-type: none"> • Clic en descargar álbum
Datos de salida esperados	Nos permite guardar el álbum con todos sus modelos y complementos en formato .rar con la misma estructura.
Datos de salida obtenidos	La salida es correcta, ya que se muestra una ventana del sistema operativo para guardar el álbum en formato .rar. Dentro se encuentra el álbum con los modelos y sus complementos.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.2.5
Nombre	Descargar todos los álbum
Requisito mínimo relacionado	RF-13
Datos de entrada	<ul style="list-style-type: none"> • Clic en descargar todos los álbumes
Datos de salida esperados	Nos permite guardar todos los álbumes con todos sus modelos y complementos en formato .rar con la misma estructura.
Datos de salida obtenidos	La salida es correcta, ya que se muestra una ventana del sistema operativo para guardar el álbum en formato .rar. Dentro se encuentra el álbum con los modelos y sus complementos.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.3.1
Nombre	Insertar Complemento
Requisito mínimo relacionado	RF-14
Datos de entrada	<ul style="list-style-type: none"> • Nombre: Dibujo • Descripción: Dibujo del modelo v1.0 • Complemento: tamaño - 163KB
Datos de salida esperados	Crea un registro del complemento con los datos introducidos y redirección a la página del modelo. Automáticamente se le asigna el complemento al modelo que hemos seleccionado previamente.
Datos de salida obtenidos	Inserta correctamente el complemento en la base de datos y guarda el archivo en el servidor.
Tiempo de procesamiento.	1 segundos
Comentario	El tiempo es despreciable siempre y cuando el peso de la imagen es menor de 1 MB.

Referencia	P-2.3.2
Nombre	Editar complemento
Requisito mínimo relacionado	RF-15
Datos de entrada	<ul style="list-style-type: none"> Nombre: Complemento Y Descripción: Texto que describe el complemento Complemento: tamaño - 23KB
Datos de salida esperados	Se cambia el registro de la base de datos y además se borra el archivo anterior para sustituirle por el nuevo. Se redirección a la vista del modelo que contiene este complemento modificado.
Datos de salida obtenidos	Hace correctamente lo esperado.
Tiempo de procesamiento.	1 segundos
Comentario	Ninguno

Referencia	P-2.3.3
Nombre	Suprimir complemento
Requisito mínimo relacionado	RF-16
Datos de entrada	<ul style="list-style-type: none"> Clic en el link de suprimir complemento
Datos de salida esperados	Al realizar la acción se debe eliminar del servidor el archivo del complemento y borrar el registro del complemento. El sistema redireccionado a la vista del modelo que se a borrado el complemento.
Datos de salida obtenidos	Realiza la operación según lo esperado.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.3.4
Nombre	Descargar un complemento
Requisito mínimo relacionado	RF-17
Datos de entrada	<ul style="list-style-type: none"> Clic en descargar complemento
Datos de salida esperados	Nos permite guardar el complemento en formato .rar .
Datos de salida obtenidos	La salida es correcta, ya que se muestra una ventana del sistema operativo para guardar el complemento.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2.4.1
Nombre	Pedir un modelo
Requisito mínimo relacionado	RF-18
Datos de entrada	<ul style="list-style-type: none"> Click en el modelo que quieres pedir Mensaje: Hola me gustaria ese modelo
Datos de salida esperados	Guarda un registro en la base de datos con los datos de la petición y muestra un mensaje de se ha realizado correctamente la petición. Redirecciona a peticiones enviadas.
Datos de salida obtenidos	La salida obtenida es la deseada
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.4.2
Nombre	Pedir un modelo rellenando los datos previamente
Requisito mínimo relacionado	RF-19
Datos de entrada	<ul style="list-style-type: none"> Clic en el modelo que quieres Clic en rellenar previamente. Seleccionar: Sin clasificar Nombre: Armario
Datos de salida esperados	Guarda un registro en la base de datos con los datos de la petición y un modelo predefinido. Muestra un mensaje como que se ha realizado correctamente la petición. Redirecciona a la vista peticiones enviadas.
Datos de salida obtenidos	La salida obtenida es la deseada
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.4.3
Nombre	Rechazar una petición
Requisito mínimo relacionado	RF-20
Datos de entrada	<ul style="list-style-type: none"> Clic en rechazar
Datos de salida esperados	Modifica el registro de la petición y lo pone como rechazado. Muestra un mensaje como la acción se ha realizado correctamente.
Datos de salida obtenidos	Obtenemos la salida esperada tal y como está definida.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.4.4
Nombre	Aceptar petición
Requisito mínimo relacionado	RF-21
Datos de entrada	<ul style="list-style-type: none"> Clic en aceptar
Datos de salida esperados	Modifica el registro de la petición y lo pone como aceptado. Se realiza una copia del modelo para el que lo pidió. Muestra un mensaje como de se ha realizado correctamente.
Datos de salida obtenidos	Hace correctamente lo que esperaba.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.5.1
Nombre	Editar perfil
Requisito mínimo relacionado	RF-22
Datos de entrada	<ul style="list-style-type: none"> Nombre: Rubén Apellidos: Martín Martín Fecha de nacimiento: 10/05/1988 teléfono: 646224640
Datos de salida esperados	Guarda los datos del perfil y los muestra.
Datos de salida obtenidos	El perfil es editado correctamente.
Tiempo de procesamiento.	Irrelevante
Comentario	En el campo de nacimiento empieza por 2020 y es poco útil. Debemos cambiar ese valor por defecto.

Referencia	P-2.5.2
Nombre	Cambiar email
Requisito mínimo relacionado	RF-22
Datos de entrada	<ul style="list-style-type: none"> Email: ivan3@gmail.com
Datos de salida esperados	Se envía un correo al email para validar el cliente. Se redirección al panel de control de ARkivia con un mensaje de que se ha realizado correctamente el envío del email.
Datos de salida obtenidos	No se envía el email al usuario.
Tiempo de procesamiento.	Irrelevante
Comentario	Este error es debido a que el servidor de correos está mal configurado.

Referencia	P-2.5.3
Nombre	Cambiar contraseña
Requisito mínimo relacionado	RF-22
Datos de entrada	<ul style="list-style-type: none"> Contraseña: isengard34
Datos de salida esperados	Se envía un correo al email para validar el cliente. Se redirección al panel de control de ARkivia con un mensaje de que se ha realizado correctamente el envío del email.
Datos de salida obtenidos	No se envía el email al usuario.
Tiempo de procesamiento.	Irrelevante
Comentario	Este error es debido a que el servidor de correos está mal configurado.

Referencia	P-2.5.4
Nombre	Borrar perfil donando los modelos
Requisito mínimo relacionado	RF-25
Datos de entrada	<ul style="list-style-type: none"> Clic en el botón de borrar
Datos de salida esperados	Borra todas sus datos como cliente y los modelos pasan a ser del usuario ARkivia.
Datos de salida obtenidos	El sistema hace correctamente lo que esperaba.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.5.5
Nombre	Borrar perfil sin donar los modelos
Requisito mínimo relacionado	RF-25
Datos de entrada	<ul style="list-style-type: none"> Clic en el botón de borrar
Datos de salida esperados	Borra el registro del usuario como los de todos sus modelos, eliminando también del servidor todos los archivos de modelo.
Datos de salida obtenidos	Hace correctamente lo que esperaba.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.5.6
Nombre	Borrar perfil donando los modelos públicos
Requisito mínimo relacionado	RF-25
Datos de entrada	<ul style="list-style-type: none"> • Clic en el botón de borrar
Datos de salida esperados	Borra el registro del usuario como los de todos sus modelos privados, eliminando también del servidor los archivos de modelo. Los modelos públicos pasan a ser donados a la cuenta de ARkivia.
Datos de salida obtenidos	El sistema realiza esta prueba según lo esperado.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-2.6
Nombre	Logout
Requisito mínimo relacionado	RF-24
Datos de entrada	<ul style="list-style-type: none"> • Clic en logout
Datos de salida esperados	Se cierra la sesión y redirección a la página principal con un mensaje de despedida.
Datos de salida obtenidos	El sistema lo hace correctamente.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-3.1
Nombre	Comprobación de usuario y contraseñas
Requisito mínimo relacionado	Comunicación entre aplicaciones
Datos de entrada	<ul style="list-style-type: none"> • Usuario: sandra22@gmail.com • Password: El código que se genera para las contraseñas.
Datos de salida esperados	En formato JSON el ID del usuario
Datos de salida obtenidos	El sistema muestra en formato JSON la información deseada
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-3.2
Nombre	Comprobación de usuario y contraseñas
Requisito mínimo relacionado	Comunicación entre aplicaciones
Datos de entrada	<ul style="list-style-type: none"> • Id del usuario: 45
Datos de salida esperados	Información en formato JSON de la jerarquía de los álbum y los modelos del usuario
Datos de salida obtenidos	El sistema muestra en formato JSON la información deseada.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

Referencia	P-3.3 Devolver el modelo
Nombre	Comprobación de usuario y contraseñas
Requisito mínimo relacionado	Comunicación entre aplicaciones
Datos de entrada	<ul style="list-style-type: none"> • Id del usuario: 45 • Id del modelo: 23
Datos de salida esperados	Información en formato JSON del modelo del usuario
Datos de salida obtenidos	El sistema muestra en formato JSON la información deseada.
Tiempo de procesamiento.	Irrelevante
Comentario	Ninguno

6.3 Resultados de las pruebas

En este apartado trataremos de interpretar los resultados que hemos obtenido tras la primera vez que realizamos la totalidad de la batería de pruebas.

La primera prueba que no me ha dado el resultado esperado ha sido el 1.1 “*formulario de contacto*”. Debido a una mala configuración del servidor de correos, por lo que se ha decidido de momento no enviar mail y hacerlo mediante un clic en un link tras realizar el login correctamente.

Este error repercute en las pruebas 1.1.3, 2.5.1 y 2.5.2 que no permiten enviar un email al correo de soporte de ARkivia. Debido a que no es un proceso de prioridad alta, se desestima su resolución actualmente.

La siguiente prueba que no ha sido satisfactoria a sido 2.1.8, la cual no almacena el modelo en la carpeta sin clasificar. Esto es debido a que en el registro no se creaba la carpeta “Sin registro” por defecto, por lo que se ha modificado el código.

6.4 Conclusiones

En esta sección se han implementado las pruebas necesarias para probar el sistema y ver que funciona como se desea. Al tener que pensar una secuencia hace que repasó todo el proyecto y así se puede ver que cada caso de uso está contemplado. De esta forma no nos olvidamos de nada y hacemos las pruebas de cada una de las acciones que nos pida el cliente.

Teniendo una metodología a seguir esto hace que vaya más rápido y hagamos las pruebas de forma ordenada comentando cada uno de los casos y guardando la información obtenido. El apartado que más información nos da es la interpretación de los resultados y que solución debemos aplicar a cada una de las pruebas que no han salido bien o que han tardado demasiado.

La sección de API es muy importante hacer las pruebas oportunas ya que otra aplicación depende de esta. Si existen errores en esta parte, que no ha sido el caso, se tendría que dar máxima prioridad ya que la otra aplicación se alimenta de los datos que le sirve este servicio web.

7 ARkivia viewer

En esta sección hablaremos sobre la aplicación móvil que tenemos que realizar. Utilizaremos el mismo esquema que la aplicación web, con los apartados de análisis, diseño, implementación y pruebas de validación para completar el desarrollo de esta aplicación. En algunos casos reutilizamos el trabajo hecho anteriormente, como por ejemplo en el análisis de los usuarios.

Hemos utilizado como fuentes e inspiración dos proyectos de la Universidad de la Valladolid de aplicaciones Android. (35; 36)

7.1 Análisis

En este apartado hemos analizado las necesidades del usuario para poder hacer un producto que satisfaga sus necesidades. Una vez sacada la información del usuario, ya nos podemos hacer una idea de cómo quiere interactuar con el producto y cuáles son sus necesidades reales. Tras el estudio del cliente debemos definir todos los casos de uso que vamos a tratar en la aplicación y definir todos los requisitos funcionales y no funcionales que va a tener la aplicación móvil.

La primera tarea que realizamos para el análisis es la definición de usuarios objetivos. En este caso tenemos dos usuarios bien diferenciados. Los usuarios que vienen de la aplicación web y quieren ver sus modelos con la aplicación, y por otro lado están los usuarios que se han descargado la app y quieren una experiencia con la realidad aumentada. El objetivo es satisfacer a los dos usuarios con la aplicación móvil que realicemos.

Los perfiles que se pueden sacar de los usuarios de esta aplicación móvil son muy parecidos a la aplicación web, donde existe un perfil más técnico que utiliza la aplicación con sus propios modelos y no buscar nuevos conocimientos, sino la solución a sus problemas. El otro perfil es el que busca una experiencia y/o aprender algo nuevo, en este caso la realidad aumentada o la visualización de modelos.

En este momento es preciso analizar el mercado de aplicaciones móviles con realidad aumentada. Para mi proyecto es necesario un cliente de realidad aumentada para poder visualizar de forma virtual nuestro modelo. Una opción es integrar un cliente dentro de nuestra aplicación para poder utilizarlo sin necesidad de una aplicación externa. Por ello vamos a hacer un estudio del mercado de aplicaciones de realidad aumentada existentes en el Play Store de Google.

- 3D Model view: Permite los formatos .off y .obj y la aplicación ocupa 5,8 MB. Quizás sea una de las mejores opciones si trabajamos estos formatos. Es una aplicación gratuita y su última actualización es del 2011, lo que nos hace pensar que está abandonada.
- HD Model Viewer: Esta aplicación permite muchos más formatos que la anterior, además de la inserción de texturas. Permite conectarse a un servidor para bajar los modelos que necesitamos y cargar modelos que estén en nuestro Smartphone. Podemos decir que es la más completa de las aplicaciones que vamos a ver en este apartado, además tiene una calidad alta. Ocupa 13MB y la última actualización es del 2015.

- AndAR Model Viewer: Este visualizador de modelos 3D nos permite cargar nuestros modelos desde la SD o dispositivo de almacenamiento que tengamos. No es muy estable con sus propios modelos, lo que nos hace dudar sobre otros modelos más pesados. Esta aplicación es más bien para iniciarse y hacer pruebas con la realidad aumentada. Su tamaño es de 846 KB y su última actualización es del 25 de agosto de 2013.
- Mobile Model Viewer: Esta aplicación nos permite cargar nuestros modelos desde la SD. Utiliza solo formatos .dae de Collada. Su tamaño es de 6MB y su última actualización es del 12 de noviembre de 2013.

En conclusión, cualquiera de los clientes de realidad aumentada es válido siempre que nos permita visualizar los modelos de ARkivia. Dependiendo de la calidad o de los formatos con los que trabajamos es mejor renderizar con uno u otro. Sin duda HD Model Viewer y Mobile Model Viewer son las mejores opciones, pero su tamaño de 13 MB y la restricción de formato a .dae, respectivamente, son los puntos negativos de estas aplicaciones.

Debido al análisis y las conclusiones obtenidas debemos valorar la opción de utilizar una librería dentro de la aplicación móvil para la realidad aumentada. Hemos realizado un estudio de las diferentes librerías (37) y al final nos decantamos por el proyecto AndAR por su documentación, su reducido tamaño y que además permite la utilización de los modelos en formato obj.

Una vez llegados a este punto, definimos las tareas que son necesarias para satisfacer a los usuarios. Son cuatro las tareas principales:

La primera es realizar un área de usuario donde se puedan ver los modelos que tienen en la aplicación de ARkivia. Se da la posibilidad de descargarse al móvil todos sus modelos así como eliminarlos.

La segunda tarea sería la implementación de un servicio de realidad aumentada para la visualización de modelos.

La tercera tarea tiene que ver con la inserción de un visor de 3 dimensiones para poder ver los modelos sin necesidad de tener un marker.

La última tarea será crear un área de prueba para los que no tienen cuenta en ARkivia y desean utilizar la realidad aumentada.

Una vez definidas las tareas y ver cuáles son las necesidades, tenemos que definir el modelo de dominio. En mi caso, este modelo solo tiene una entidad, la de modelo. No es necesario definir más entidades ya que toda la información se pide al servidor de ARkivia, mediante el protocolo API REST, por lo que no es necesario volcar todos los datos a la aplicación móvil y tener un modelo de dominio al de la aplicación web.

A continuación definiremos el comportamiento del sistema y todas sus funcionalidades básicas. La aplicación ofrece esta lista de funcionalidades:

- RF-1: Autenticarse.
- RF-2: Sincronizar los álbumes.
- RF-3: Descargar los modelos.
- RF-4: Suprimir los modelos.

- RF-5: Ver modelo en 3D.
- RF-6: Probar la realidad aumentada.

Este sistema tiene estas funcionalidades pero debemos definir cuáles son los requisitos no funcionales que debemos cumplir. Al ser una aplicación móvil los requisitos no funcionales son diferentes a una aplicación web.

La aplicación deberá poder instalarse en un 75% de terminales actualmente en el mercado. No sobrepasará los 20MB una vez instalado en el dispositivo, sin contar con los modelos, únicamente el programa. Los tiempos de carga de una vista a otra no excederán los dos segundos.

Las descargas de los archivos de modelos no sobre pasarán el minutos para los archivos de tamaño menor que 1 MB. Si en la carga de la vista implica una o varias peticiones al servidor, no excederá más de cinco segundos.

Las navegación debe de ser intuitiva y las vistas amigables. Deberá al menos estar en los idiomas castellano e inglés.

El diagrama de casos de uso tras haber analizado por completo se muestra en la figura 35.

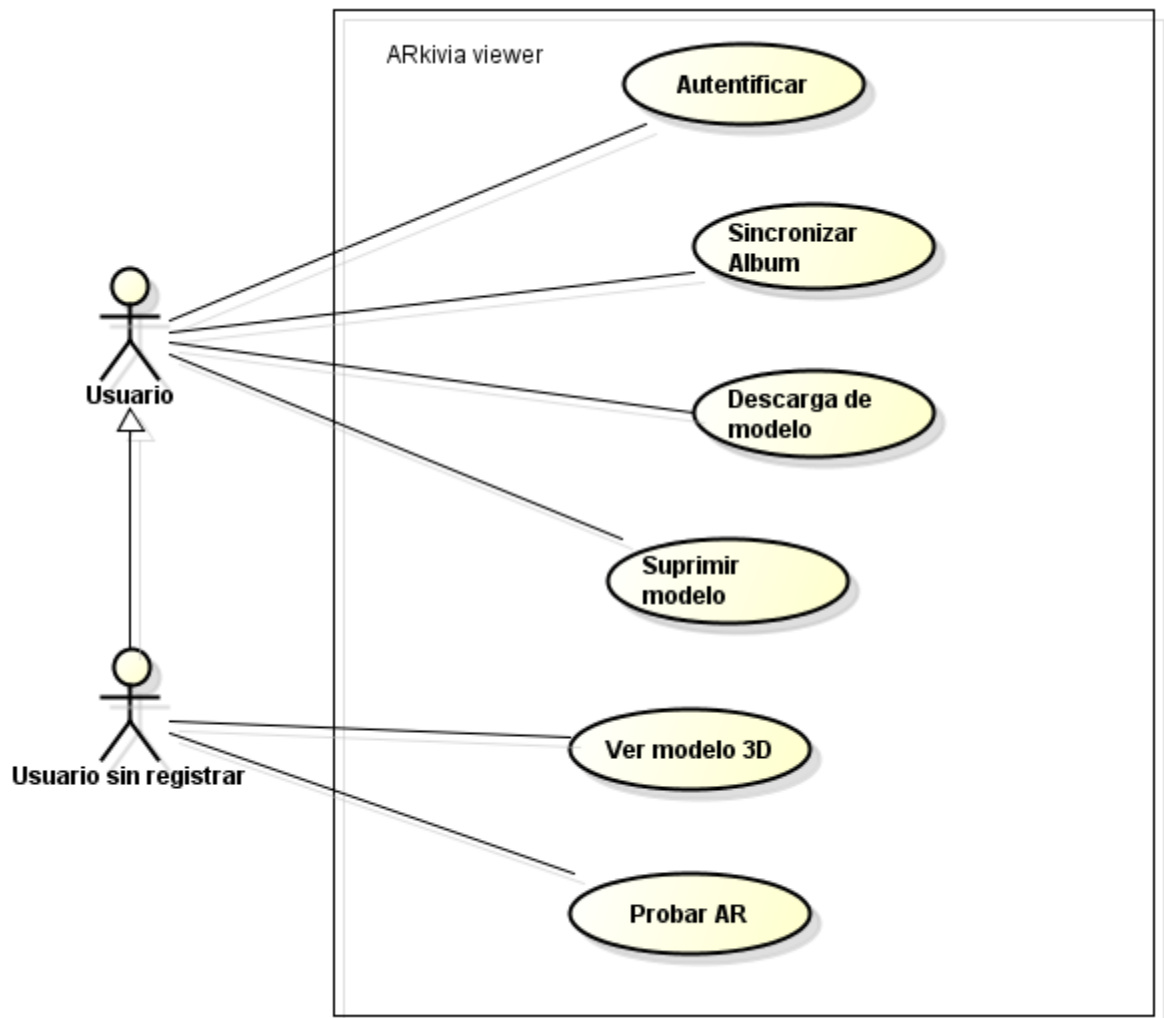


Figura 35 - Diagrama de casos de uso de la aplicación de ARkivia Viewer.

En este caso los actores son los mismos que en la aplicación web, por lo que no vuelvo a definirlos.

Para cada caso de uso del diagrama vamos a especificar una descripción, la entrada de datos, precondiciones, la secuencia del caso de uso, las postcondiciones, las excepciones y una nota si fuera necesario.

CU-1	Autenticar
Descripción	El actor ACT-1 quiere loguearse.
Entrada	<ul style="list-style-type: none"> Email: sandra@gmail.com Password: lisboa12
Precondiciones	Haber realizado el caso de uso CU-4 de la aplicación web
Secuencia	<ol style="list-style-type: none"> 1. El actor pulsa el botón de loguearse 2. El sistema envía una petición REST al API de ARkivia, espera la respuesta y muestra el área de usuario.
Postcondiciones	Ninguna
Excepciones	2. Si el sistema detecta un error vuelve al paso 1.
Nota	Ninguna

CU-2	Sincronizar álbum
Descripción	El actor ACT-1 desea ver la jerarquía de sus álbumes para buscar un modelo
Entrada	Ninguna
Precondiciones	CU-1 de la aplicación web.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic en el botón de ver álbum. 2. El sistema envía una petición REST al API de ARkivia, espera la respuesta y la jerarquía de álbumes con los modelos que contiene.
Postcondiciones	Ninguna
Excepciones	2. Si el sistema detecta un error vuelve al paso 1.
Nota	Ninguna

CU-3	Descarga de modelo
Descripción	El actor ACT-1 quiere descargarse al móvil un modelo
Entrada	Ninguna
Precondiciones	Haber realizado previamente CU-5 en la aplicación web y CU1 de la aplicación web.
Secuencia	<ol style="list-style-type: none"> 1. El actor hace clic al botón de descargar modelo. 2. El sistema envía una petición REST al API de ARkivia, espera la respuesta, crea el archivo del modelo en el dispositivo y muestra un mensaje como que se ha realizado con éxito..
Postcondiciones	Ninguna
Excepciones	2. Si el sistema detecta un error vuelve al paso 1.
Nota	Ninguna

CU-4	Suprimir modelo
Descripción	El actor ACT-1 desea eliminar el modelo en su dispositivo.
Entrada	Ninguna
Precondiciones	CU-3
Secuencia	1. El actor hace clic en el botón de suprimir. 2. El sistema elimina el archivo del modelo del móvil.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Al suprimir el modelo no se elimina de ARkivia sino únicamente el archivo alojado en el móvil.

CU-5	Ver modelo en 3D
Descripción	El actor ACT-1 o ACT-2 quiere ver un modelo en 3D.
Entrada	Ninguna
Precondiciones	Ninguna
Secuencia	1. El actor pulsa el botón de ver modelo en 3D. 2. El sistema carga el modelo y muestra el modelo en pantalla.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

CU-6	Probar la realidad aumentada
Descripción	El actor ACT-1 o ACT-2 quiere ver un modelo con realidad aumentada.
Entrada	Ninguna
Precondiciones	Ninguna
Secuencia	El actor pulsa el botón de probar realidad aumentada. El sistema carga el modelo y muestra por pantalla la cámara del dispositivo.
Postcondiciones	Ninguna
Excepciones	Ninguna
Nota	Ninguna

A la hora de desarrollar el prototipo y posteriormente enseñárselo al cliente, en este caso el tutor del proyecto, se detectaron dos carencias. La primera es la necesidad de visualizar los modelos más vistos en la primera vista, tras realizar el logueo. La segunda ha sido la falta de un botón de recordar contraseña. Estas dos utilidades nuevas no se han tomado como nuevos casos de uso.

7.2 Diseño

Este apartado va únicamente referido al diseño de la aplicación móvil. Este diseño está basado en el análisis que se ha realizado anteriormente con el usuario.

La tecnología que utilizamos para la implementación en la aplicación móvil es el sistema Android. No realizamos un estudio de los posibles sistemas, ya que se ha impuesto este sistema para la realización del proyecto. No obstante, sería interesante, en un futuro, analizar cuál de los sistemas que nos ofrece el mercado es el más idóneo para nuestro producto.

La arquitectura del sistema viene definida por el propio sistema Android el cual sigue este esquema que vemos en la figura 36.

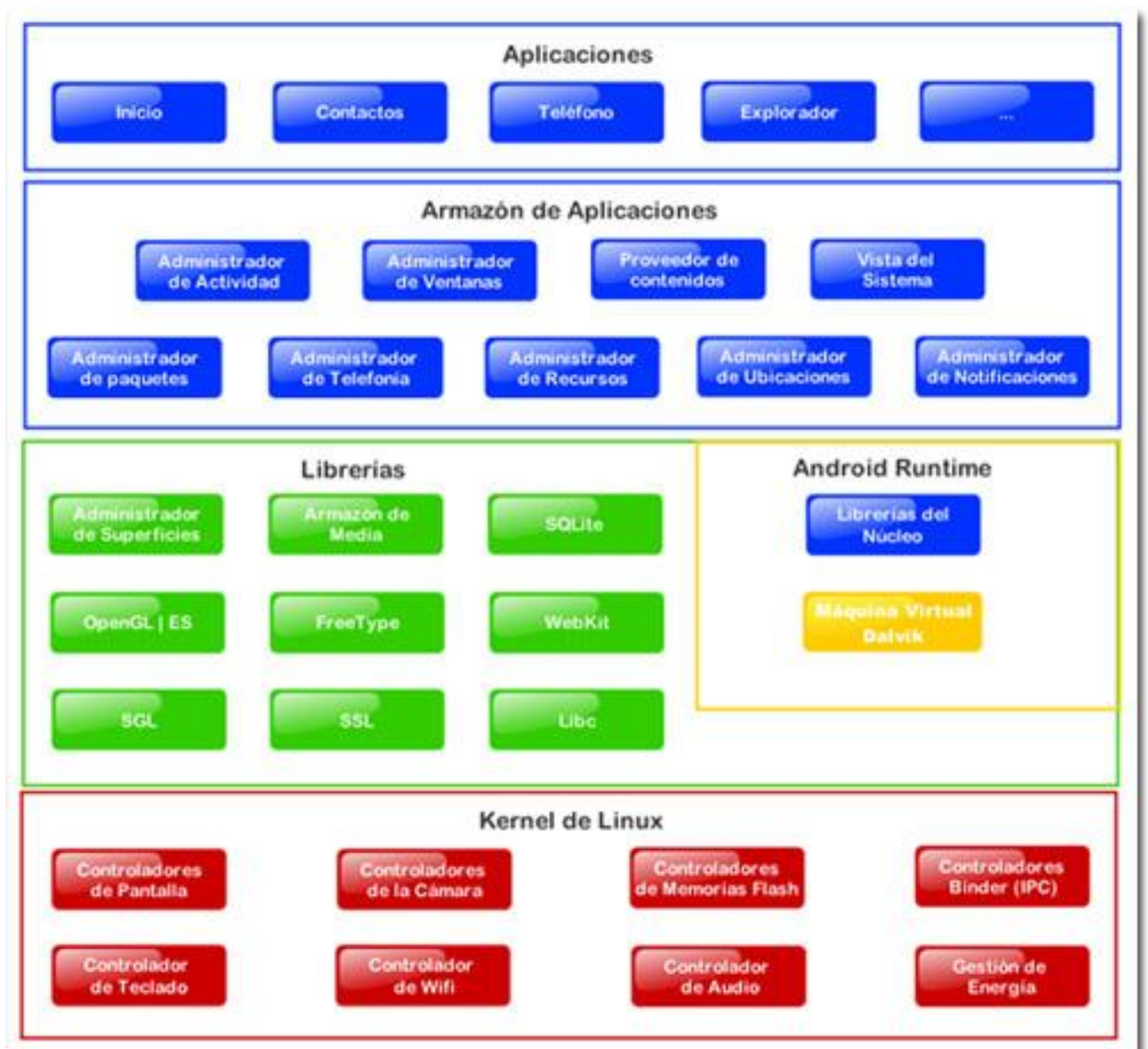


Figura 36 - Representación de la arquitectura de Android. (38)

Es importante saber que esta es la arquitectura fuera de la aplicación, la cual estamos obligados a seguir pero dentro de la capa de aplicaciones nosotros somos los que decidimos la arquitectura de la aplicación. El patrón de diseño que vamos a seguir es el MVVM (Model-View-ViewModel) que es uno de los más popular y utilizado en aplicaciones Android. Este patrón consiste en tres capas que definimos a continuación:

- **Modelo:** Es la capa que se encarga de los modelos definidos en el sistema. En este caso tenemos uno definido en el modelo de dominio, pero más adelante veremos la necesidad de definir otro modelo, el de álbum.
- **Vista:** Esta capa, es donde se alojan los archivos de interfaz de usuario, en nuestro caso, en formato XML. Estos archivos contienen la estructura de los elementos que se ven por pantalla.
- **Vista Modelo:** Esta capa alberga las clases que se encargan de unir la capa de vista y de modelo. Se encarga de la interacción que hace el usuario con la aplicación. Es donde se gestiona la lógica de la aplicación y se encarga de recoger los eventos de la vista para realizar las acciones necesarias con el modelo. Las clases de este tipo extienden de una clase llamada Activity. Está orientada a eventos ya que se disparan eventos en cada acción que realiza el usuario y se actúa en los casos definidos.

Este patrón me ayudará a conseguir una aplicación bien estructurada, fácil de mantener y escalable.

A continuación vamos a diseñar la base de datos para poder guardar cierta información en la aplicación. Es necesario hacerse una pregunta ¿Por qué no se guarda toda la información del sistema de ARkivia web, en la aplicación móvil? Porque no es necesario, ya que el usuario de la aplicación móvil sólo usa datos de su usuario y no existe la necesidad de descargarse información de terceros. Sería ineficiente hacer una petición de toda la base de datos y actualizarla cada vez que el usuario entre en la aplicación. Hay que tener presente que al realizar un API REST también estamos aumentando los puntos críticos del sistema donde un ataque a la aplicación pondría en riesgo el sistema.

Con las peticiones REST al servidor de ARkivia sólo recibimos la información necesaria que pide el usuario en ese momento sin la necesidad de enviar toda la información extra que no se va a utilizar. La primera petición que se hace siempre es autenticarse en ARkivia, para así lograr el id del usuario y poder identificarse las siguientes peticiones. Este id del usuario es la forma de decir que está autenticado porque conozco mi id de usuario. Después realizo el envío de las peticiones de jerarquía de álbumes y modelos, o de descarga un modelo con este id.

Dicho esto, solo es necesario conservar el id del usuario, que es el mismo que el del sistema ARkivia, y los datos del modelo, como el nombre y la posición en el ranking de visualización. Estos son los datos que vamos a guardar en la base de datos.

A continuación, muestro la clase de modelo, con sus atributos y métodos implementados en la figura 37.

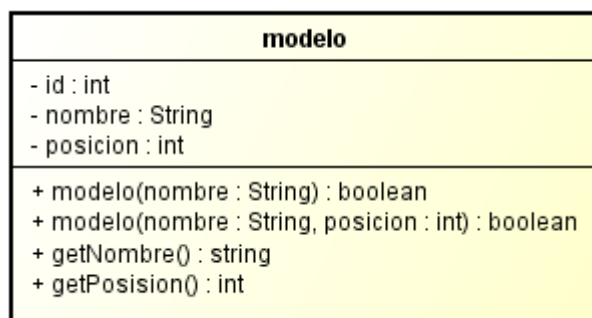


Figura 37 - Diagrama de clases de la aplicación móvil.

Veremos más adelante porque es necesario definir el atributo posición. Con este modelo cubrimos las necesidades del sistema que hemos implementado.

Utilizamos el sistema de gestión de base de datos relacional SQLite para esta aplicación. Las características que nos han hecho decantarnos por este sistema son su fácil configuración, un único archivo de tamaño pequeño (500KiB) para su funcionamiento, es autónomo y cumple con la regla ACID (*atómica, consistente, aislada y Durable*). (39)

La estructura que tiene la aplicación la hemos dividido en tres carpetas. La primera carpeta, la llamamos ARkivia y es donde se alojan las clases relacionadas con la aplicación propia. La segunda carpeta se llama edu.dhbw donde está el proyecto de realidad aumentada de AndAR.Y la tercera carpeta es la de ivi.dhbw.objLoader que contiene los archivo para el visualizador 3D. En la figura 38 observamos esta estructura:

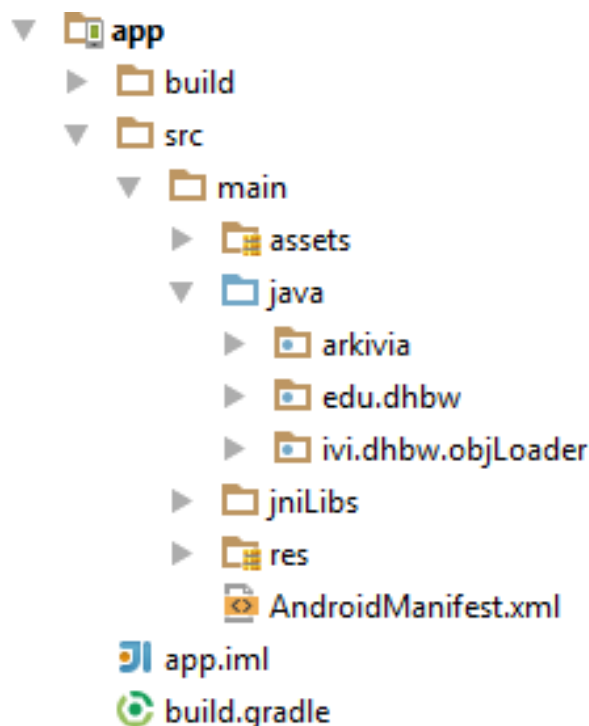


Figura 38 - Estructura general de la aplicación móvil.

Carpeta arkivia

En este directorio se encuentra las clases que implementan la aplicación en general, menos los casos de uso CU-5 y CU-6. Existen varios tipos de clase, pero el patrón que sigo en general es el MVVM. En la figura 39 vemos todas las clases que he implementado para el funcionamiento de la aplicación.

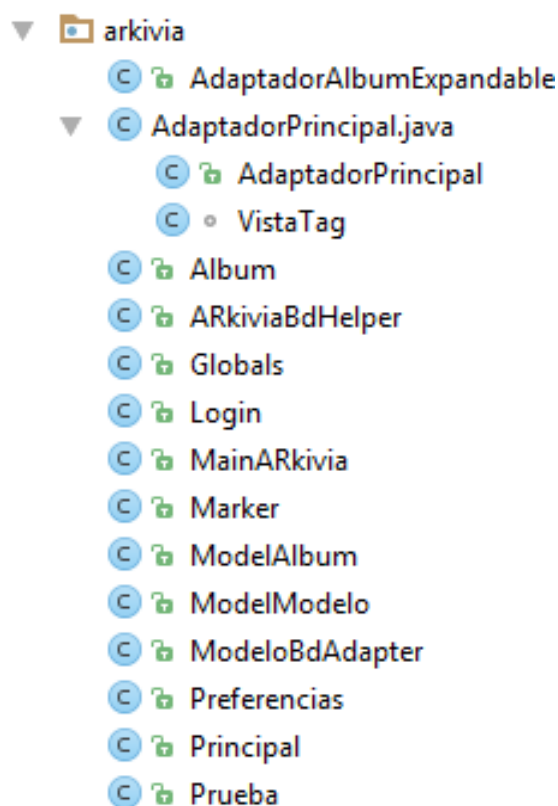


Figura 39 - Estructura de la aplicación de ARKivia propia.

Las clases de la capa de modelo son ModelAlbum y ModelModelo. La primera clase modela los álbumes, que anteriormente no vimos la necesidad de definirla en el modelo de dominio. ¿Por qué no está en el modelo de dominio? Porque no se necesita guardar la información de ella en la base de datos de la aplicación. ¿Y por qué ahora sí tenemos que hacer el modelo? Porque para la tarea de visualizar la jerarquía de álbum y modelos, utilizamos peticiones REST para obtener esa información, sin necesidad de guardarla en la base de datos, por lo que los álbumes no los guardamos, pero sí que lo utilizamos como clase para manejarlo como un objeto. Si en un futuro es necesario guardar en la base de datos los álbumes ya tendríamos el modelo hecho.

¿Por qué el modelModelo si está en el modelo de dominio? Porque necesito saber con qué nombre se ha guardado en el dispositivo y saber la posición que ocupa en el ranking de visualizado en el área de usuarios. Estos atributos son específicos de la aplicación móvil por eso se guardan en la base de datos y no se envían a la aplicación web.

Las clases de la capa vista-modelo son las que extienden de la clase padre Activity y son las encargadas de captar los eventos que ocurren en la interfaz. Prueba, Principal, Preferencias, Marker, MainARKiva, Login y Album. Las clases de álbum y principal son las dos que son únicamente para usuarios registrados de ARKivia. Las demás vistas las pueden ver todas las personas que descarguen la aplicación.

Los elementos que se ven en la vista son definidos en los archivos XML. En la figura 40 muestro todas las vistas de la aplicación:

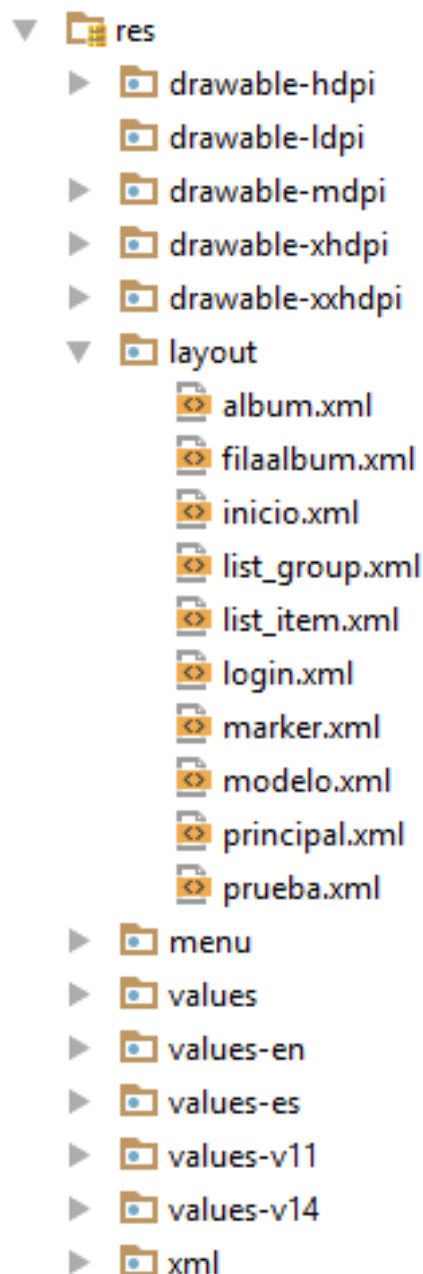


Figura 40 - Estructura de los recursos de la aplicación de ARkivia Viewer.

En la carpeta “*menu*” que está debajo de “*layout*”, están los archivos XML que definen la estructura del menú superior de las distintas vistas.

Otras de las clases de vista-modelo de la carpeta “*arkivia*” son *AdaptadorAlbumExpandable* y *AdaptadorPrincipal*. Estas clases están para asistir a la clase de la vista porque necesitan de un modelo más sofisticado y con mayor especialización. En este caso es para las listas de ranking de modelos vistos, la visualización por listas y desplegaables de la jerarquía de álbumes con sus modelos.

Existen también otro tipo de clase que no se contemplan en el patrón MVVM que son:

- La clase *Globals*, la utilizó para variables globales como la dirección del servidor o el sistema de salir de la aplicación que utilizo. Si la variable *fin* tiene valor 0, la aplicación sigue su flujo normal. En caso de que no sea 0 el valor de *fin*, cada *Activity* que esté abierto se manda que se cierre al recibir la acción de mostrarse.

- La clase ARkiviaBdHelper crea la tabla de modelo en base de datos la primera vez que se necesita.
- La clase ModeloBDAdapter es la encargada de sacar objetos de los modelos de la base de datos. Tiene como métodos existeModelo(), borrarModeloId(), insertarModelo(), ponerPrimero(), getModelosUltimos(), getModelos() y borrarTodo().

Estas son las clases que gestionan la aplicación, existen otros elementos como las imágenes utilizadas, los valores generales, la configuración aplicada y la traducción en inglés en la carpeta *res*, que muestro en el apartado de layout.

Para finalizar el apartado de diseño, es necesaria la definición de la navegación en la aplicación. Para ello, al igual que en la aplicación web hemos esquematizado las diferentes vistas que puede tener la aplicación y cuál sería su navegación idónea.

En la figura 41 se puede ver el workflow de la aplicación:

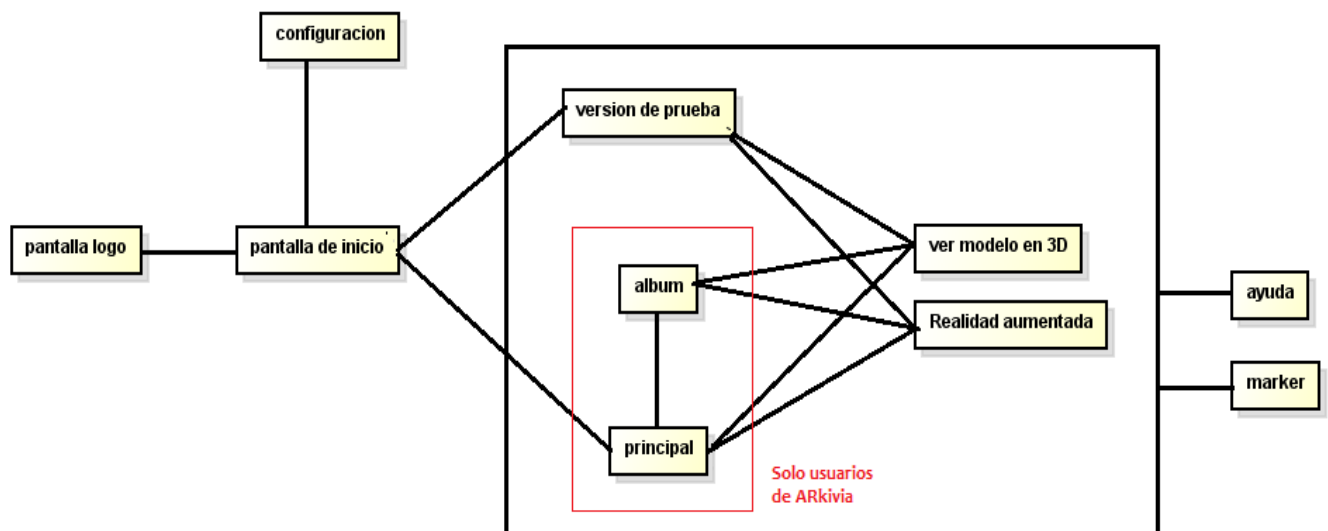


Figura 41 - Workflow de la aplicación web.

Como podemos ver en el esquema se reutilizan vistas como la de ver el modelo en 3D y la de realidad aumentada, para no tener que crear para cada flujo una nueva vista e implementación. También se pueden acceder a la ayuda y al marker para la realidad aumentada en el resto de las vistas que no sean configuración, pantalla de logo o pantalla de inicio. Se puede ver también como en el área de usuario de ARkivia, que solo se puede acceder en caso de haberse autenticado correctamente. Tras esto se encuentran las vista de principal y álbum donde se le mostrará los modelos que tiene en ARkivia.

Tras realizar este diseño se implementó un prototipo que simplemente navegaba entre las diferentes vistas. Esto nos permitió validar la navegación con el cliente y tener la primer realimentación de la aplicación móvil. Tuvimos que modificar el área del usuario ya que requería tener un ranking de los modelos que había visto con anterioridad y tuviese descargados en el smartphone. Esto le sirve al usuario para no tener que buscar en la jerarquía de álbum y ser más rápido al mostrar el modelo.

Se han realizado los diagramas de secuencia de los casos de uso principales, para poder ver la interacción de los modelos. Uno de los casos de uso está representado en las ilustraciones 7, 8 y 9 en el anexo de diagramas. Los demás diagramas de secuencia de la aplicación móvil se encuentran en el archivo .asta dentro de la carpeta *desarrollo*, en el CD-ROM adjunto a esta memoria.

7.3 Implementación

Ahora vamos a explicar que contienen las otras dos carpetas que están en el directorio "src/main/java". La primera hace referencia al proyector AndAR y como hemos realizado su implementación. La segunda es de la librería objLoader, es el visualizador de modelos.

Carpeta edu.dhbw

En esta carpeta está el proyecto de realidad aumentada AndAR. Lo he descargado en la siguiente URL: <https://code.google.com/p/andar/wiki/AndARModelViewerInstructions> la versión más actual. Primero explicaré cómo funciona, las partes que tiene y después como lo he integrado dentro del proyecto.

Lo primero que hace el software de AndAR es pedir por parámetro la dirección donde se encuentra el archivo de .OBJ. Se lo hago llegar a través de un objeto intent de esta forma `intent.putExtra("name", "arkivia.obj");` después se encarga de cargarlo y lo transforma en un objeto `Model3D`. En este último objeto hay que saber que tiene que cargar los parámetros de textura e iluminación que se encuentra en el archivo .MTL. En el archivo .OBJ existe una referencia al archivo .MTL que se le aplica. Por lo que a la hora de generarlos, no puedo cambiar de nombre en ningún caso el archivo .MTL.

Una vez que tiene el objeto `Model3D` construido y con todos sus parámetros es cuando inicia la cámara y muestra por pantalla lo que se capta a través de ella.

El siguiente paso es el de detección del marker; analiza cada imagen de la pantalla para ver si existe el marker. No es necesario que sea el marker exacto, admite cierto margen de error. Si existe el marker se ejecuta la función de pintar el modelo, sino sigue procesando las imágenes hasta encontrar la marca. Mientras se está dibujando en segundo plano se está ejecutando el método de detección de marker, por lo que si desaparece, el método de dibujar el modelo deja de ejecutarse.

Para que el AndAR funcione debo pasarle la dirección donde está el archivo del modelo en el formato .OBJ para que lo cargue y pueda imprimir por pantalla. Hay que tener especial cuidado en alojar junto al archivo .OBJ el archivo con formato .MTL que contiene la iluminación del objeto, así como las texturas.

Hemos tenido que hacer una modificación en la clase `Model3D.java` añadiendo al principio del método `draw` la función `init(gl);`. Esto se debe a que no se mostraba ningún modelo cuando detectaba el marker y encontramos una solución iniciando el objeto `gl` de esta forma.

Este proyecto es un conjunto de clases que permite cargar un modelo de formato .OBJ para poder visualizar. Existen una gran variedad de proyectos de carga de archivo OBJ y he elegido este por su simplicidad, ya que es el que menos clases tiene y más claro está. Además permite voltear el modelo.

Su funcionamiento es mucho más sencillo que la realidad aumentada ya que solo tendremos que esperar a que se cargue el modelo. Una vez cargado el Activity se encarga de ejecutar los eventos que van realizando los usuarios y aplicando las transformaciones al modelo, para ver el movimiento del objeto.

El código que he tenido que implementar ha sido el envío de la dirección donde se encuentra el archivo OBJ que puede ser de dos tipos, la dirección del archivo del modelo de la propia aplicación o en el dispositivo de almacenamiento. Además, he tenido que añadir las funcionalidades, que aparecen en el menú superior, en el Activity.

Una vez descrita toda la estructura de la aplicación y los proyectos externos utilizados, voy a explicar qué orden hemos seguido para realizarlo. Lo primero hemos reutilizado el prototipo sin funcionalidades que habíamos realizado para validar la navegación de la aplicación móvil con el cliente. Después, hemos ido añadiendo las funcionalidades que requerían los casos de uso del 1 al 4. A continuación, hemos embellecido la aplicación con los menús de la parte superior, las imágenes de los botones y los mensajes de espera, de éxito o de error. Para finalizar hemos integrado los dos proyectos para cumplir con los casos de uso 5 y 6.

Para la implementación de ARkivia Viewer se utilizaron los siguientes programas:

- Eclipse 4.2.2: Entorno de desarrollo para java.
- Android Developer Tools Versión 22.3.0: Plugin de Eclipse para desarrollo de aplicaciones Android.
- Android Studio: IDE para desarrollar aplicaciones Android.
- Utilizamos también las herramientas XAMPP y GIMP que nos sirven como servidor local y editor de imagen, respectivamente.

7.4 Pruebas de validación

Al igual que hemos hecho en la aplicación web, realizamos una batería de pruebas para validar el producto que hemos implementado. Este proceso nos sirve para confirmar que todos los casos de uso se han realizado correctamente y que funcionan según lo esperado.

La metodología que vamos a seguir es exactamente la misma que en las pruebas de ARkivia web, primero listamos toda la batería de prueba, luego describimos lo sucedido en cada una de las pruebas y para finalizar hacemos una conclusión de los resultados.

La lista de las pruebas que vamos a realizar es la siguiente:

1. Autenticarse
2. Sincronizar los álbumes
3. Descargar un modelo
4. Suprimir un modelo
5. Ver modelo en 3D
6. Probar la realidad aumentada

Ahora procedemos a realizar cada una de las pruebas en el orden establecido anteriormente.

Referencia	P-1
Nombre	Autenticarse
Requisito mínimo relacionado	RF-1
Datos de entrada	<ul style="list-style-type: none"> Email: sandra@gmail.com Password: Paris21
Datos de salida esperados	El sistema accede al área de usuario.
Datos de salida obtenidos	La salida es correcta y el sistema entra en el área de usuario.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-2
Nombre	Sincronizar los álbumes
Requisito mínimo relacionado	RF-2
Datos de entrada	<ul style="list-style-type: none"> Email: sandra@gmail.com Password: Paris21
Datos de salida esperados	El sistema accede al área de usuario.
Datos de salida obtenidos	La salida es correcta y el sistema entra en el área de usuario.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-3
Nombre	Descargar un modelo
Requisito mínimo relacionado	RF-3
Datos de entrada	<ul style="list-style-type: none"> Clic en el botón de descargar modelo
Datos de salida esperados	El sistema descarga el archivo en la carpeta de ARkivia
Datos de salida obtenidos	El sistema actúa correctamente y crea un archivo del modelo.
Tiempo de procesamiento.	5 segundos
Comentario	Si el archivo es más grande tarda más tiempo en hacer esta prueba ya que el servidor tiene que enviar más información y el dispositivo de recepción tiene que crear un archivo más grande.

Referencia	P-4
Nombre	Suprimir un modelo
Requisito mínimo relacionado	RF-4
Datos de entrada	<ul style="list-style-type: none"> Clic en el botón de borrar modelo
Datos de salida esperados	El sistema borra de la carpeta de ARkivia el archivo del modelo
Datos de salida obtenidos	El sistema realiza la acción según lo esperado.
Tiempo de procesamiento.	Irrelevante.
Comentario	Ninguno.

Referencia	P-5
Nombre	Ver modelo en 3D
Requisito mínimo relacionado	RF-5
Datos de entrada	<ul style="list-style-type: none"> Clic al botón de ver modelo en 3D
Datos de salida esperados	El sistema carga el modelo y lo muestra por pantalla.
Datos de salida obtenidos	La salida es la esperada.
Tiempo de procesamiento.	Tarda unos segundos según el tamaño del modelo.
Comentario	Ninguno.

Referencia	P-6
Nombre	Probar la realidad aumentada
Requisito mínimo relacionado	RF-6
Datos de entrada	<ul style="list-style-type: none"> Clic al botón de realidad aumentada
Datos de salida esperados	El sistema carga el modelo y muestra lo que visualiza la cámara. En el caso de que este el marker a la vista se muestra el modelo superpuesto.
Datos de salida obtenidos	El sistema realiza correctamente la prueba.
Tiempo de procesamiento.	Tarda unos segundos según el tamaño del modelo.
Comentario	En el caso que el modelo esté descargado esta prueba no funciona.

Tras realizar esta batería de prueba hemos detectado que se ejecutan correctamente todos los casos de uso. Se pueden mejorar quizás los tiempos de carga, en caso de tener modelos muy grandes, pero sería una tarea para más adelante.

8 Conclusión

En esta sección hablamos sobre el trabajo realizados durante este Trabajo de Fin de Grado. Además, a grandes rasgos, comentaremos las principales aportaciones y las líneas de trabajo futuro.

En el siguiente apartado hablamos sobre los objetivos cumplidos y se aporta una valoración personal del trabajo realizado.

8.1 Trabajo realizado

Uno de los objetivo principales de este proyecto es la implementación de una aplicación web que gestione los modelos 3D. Este objetivo se ha cumplido ya que los requisitos iniciales, más los impuestos por el tutor, los cuales están probados y funcionan correctamente en el servidor.

Otro de los objetivos es la creación de una aplicación móvil con realidad aumentada y que se conecta a la aplicación web a través del API. Este objetivo se ha cumplido también ya que la aplicación está operativa, la conexión al API y la realidad aumentada funciona para el caso de prueba.

Desde un punto de vista técnico, el trabajo me ha permitido conocer en profundidad diversas tecnologías de importancia en desarrollo web: HTML5, Foundation, Yii, etc... El framework Yii tiene una curva de aprendizaje relativamente larga, por lo que se han invertido 2 semanas sólo para lograr comprender las ventajas que me ofrecía ,como son la rapidez y facilidad con la que se programa.

Desde el punto de vista de la planificación del proyecto, me he dado cuenta que es muy difícil estimar el tiempo que se va a tardar en hacer una tarea y que lo mejor es dividirla en tareas más pequeñas para poder ajustarse mejor a las estimaciones iniciales.

Quizás en el planteamiento inicial debí elegir Bootstrap, en vez de Foundation, ya que su popularidad actualmente es mucho mayor y casi todos los proyectos utilizan este framework de front-end. Otro de los fallos es no haber iniciado en la versión alfa de Yii 2.0 y haberme quedado en la versión estable 1.1.13. Me hubiese gustado trabajar con el framework Symfony2 ya que actualmente estoy viendo que tiene mayor versatilidad para adaptarse a proyectos grandes y mayor potencial que Yii.

En la Universidad se suelen hacer proyectos de menor medida, por lo cual centrarse en un proyecto más grande te hace ver que son necesarias otras herramientas para tener controlado todo el sistema, como son una buena documentación y un gestor de versiones.

En conclusión, he aprendido una gran cantidad de cosas que me van a servir en el futuro. No me arrepiento de haber fallado en algunas cosas o que haya gastado más tiempo de lo estimado, por lo que creo que este proyecto me ha hecho mejor informático y me ha enseñado a enfrentarme a problemas que no había tenido nunca y voy a tener en un futuro.

8.2 Trabajo futuro

Una de las tareas que queda por realizar es la mejora de la parte de realidad aumentada para los objetos de modelos descargados de ARkivia web. Otra posible mejora de este proyecto es la implementación de nuevos tipos de archivos de modelo en los visualizadores 3D y realidad aumentada.

En la parte web se podría añadir un chat para la mayor interacción entre usuarios. Otra idea es ofrecer un sistema de gamificación a los usuarios para los que más utilicen la aplicación.

Otra de las partes fundamentales a mejorar, es el tema de seguridad. Uno de los puntos débiles es el paso de mensajes en el API. Para ello se podría implementar un sistema OAuth para la autenticación y autorización de los usuarios al usar el servicio web. Además de esto, podríamos añadir para acceder a ARkivia con las cuentas de Facebook o Google+ sin necesitar que el usuario se registre en nuestra plataforma.

Otra de las cosas que se podría haber realizado, es un estudio de viabilidad económica del proyecto, estimando los costes, los ingresos de la aplicación y analizar su rentabilidad.

En conclusión, este proyecto puede seguir evolucionando tanto en la parte web como en la parte móvil dependiendo del uso que hace el usuario de las aplicaciones. Se pueden hacer estudios de la tabla de logs implementada, para poder así analizar los usuarios y ver las partes más utilizadas y las que menos.

9 Bibliografía

1. **ABC.** abc.es. *¿Cuántas «apps» nos descargamos en España? Tecnología - Móviles - Aplicaciones.* [En línea] [Citado el: 15 de Julio de 2015.] <http://www.abc.es/tecnologia/moviles-aplicaciones/20130923/abci-apps-descargar-diario-201309231201.html>.
2. **periódico, El.** elperiodico.com. [En línea] <http://www.elperiodico.com/es/noticias/mobile-world-congress/vodafone-aplica-realidad-aumentada-para-guiar-los-discapacitados-1444762>.
3. **Innovae.** Realidad Aumentada . [En línea] [Citado el: 16 de Junio de 2015.] <http://realidadaumentada.info>.
4. **3D, Archive.** Archive 3D, sistema de gestión de base de datos. [En línea] [Citado el: 22 de Junio de 2015.] <http://archive3d.net/>.
5. **3D Xtra.** [En línea] [Citado el: 21 de Junio de 2015.] <http://www.3dxtras.com/>.
6. **3D modelo free.** [En línea] [Citado el: 22 de Junio de 2015.] <http://www.3dmodelfree.com/>.
7. **Internet Business Systems, Inc.** Sharecg. [En línea] [Citado el: 22 de Junio de 2015.] <http://www.sharecg.com/>.
8. **FlyingArchitecture.** [En línea] [Citado el: 21 de Junio de 2015.] <https://flyingarchitecture.com/> .
9. **Augment.** Augment. [En línea] [Citado el: 26 de Junio de 2015.] <http://www.augmentedev.com/>.
10. **Espinoza, José Martín Olguín.** Universidad Autónoma de Baja California. [En línea] [Citado el: 16 de Junio de 2015.] <http://yaqui.mx/uabc.mx/~molguin/as/RUP.htm>.
11. **Nacional, Instituto Geográfico.** Grupo B. Sistemas y Tecnologías de la información. [En línea] [Citado el: 16 de Junio de 2015.] www.ign.es/ign/resources/acercade/aig/b.pdf.
12. **Larman, C.** *Uml y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educación, 2013.
13. **Loja, Universidad técnica particular de.** Slideshare. [En línea] [Citado el: 19 de Junio de 2015.] <http://es.slideshare.net/lecastillox/gestion-del-riesgo>.
14. **Cázares Iturriaga, Mariana Lucía.** Slideshare. [En línea] [Citado el: 19 de Junio de 2015.] http://es.slideshare.net/jose_macias/gestin-del-riesgo-de-software.
15. **Ifimedia.** Blog Ifimedia. [En línea] [Citado el: 18 de Junio de 2015.] <http://www.ifimedia.com/cms-joomla-wordpress-vs-framework-laravel-django-ruby-on-rails>.
16. **Wikipedia,** framework con programación por capas. [En línea] [Citado el: 19 de Junio de 2015.] http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks.
17. **Software, Pivotal.** Spring. [En línea] [Citado el: 26 de Junio de 2015.] <https://spring.io/>.
18. **Joomla.** Joomla CMS. [En línea] [Citado el: 16 de Junio de 2015.] <http://www.joomla.org/>.
19. **Microsoft.** ASP.NET. [En línea] [Citado el: 16 de Junio de 2015.] <http://www.asp.net/>.

20. **Drupal.** Drupal CMS. [En línea] [Citado el: 16 de Junio de 2015.] <https://www.drupal.org/>.
21. **Hansson, David Heinemeier.** Ruby On Rails. [En línea] [Citado el: 16 de Junio de 2015.] <http://rubyonrails.org/>.
22. **Matters, Open Source.** Zend Framework. [En línea] [Citado el: 16 de Junio de 2015.] <http://framework.zend.com/>.
23. **LLC, Yii Software.** Yii Framework. [En línea] [Citado el: 16 de Junio de 2015.] <http://www.yiiframework.com/>.
24. **SensioLabs.** Symfony Framework. [En línea] [Citado el: 16 de Junio de 2015.] <https://symfony.com/>.
25. **Magneticone.** Cms2cms. [En línea] [Citado el: 19 de Junio de 2015.] <http://www.cms2cms.com/blog/typo3-to-joomla-reasons-and-ways-of-migration-infographic>.
26. **Simo, Jose Manuel Munoz.** Blog JOSEMMISMO. [En línea] [Citado el: 2 de Julio de 2015.] <http://josemmsimo.wordpress.com/2013/12/02/php-frameworks-comparison-zend-symfony-codeigniter-yii-and-cake-php/>.
27. **BPlusPlus.** briananglin. [En línea] [Citado el: 17 de Junio de 2015.] <http://briananglin.me/2014/01/best-php-frameworks-2014/>.
28. **ZURB.** Zurb foundation. [En línea] [Citado el: 16 de Junio de 2015.] <http://foundation.zurb.com/>.
29. Wikipedia. *Programación por capas*. [En línea] [Citado el: 18 de Junio de 2015.] http://es.wikipedia.org/wiki/Programación_por_capas.
30. **Blanco Abal, Josefa y Maroto Garzón, Nuria.** " PYMEMÓVIL " *Aplicación web y tienda online de telefonía*. E.U. de Informática (Segovia) : Universidad de Valladolid, 2013.
31. **Junquera Gómez, Mario y Crespo, Yania.** *Gestión automatizada de la Escuela de Animación de Scouts de Castilla y León*. Proyecto Fin de Carrera E.T.S.I. Informática : Universidad de Valladolid, julio 2013.
32. **Santiago de la Parte Flórez y la Fuente Redondo, Pablo de.** *Aplicación Web para la visualización y consulta del Padrón Municipal de Valladolid*. Trabajo Fin de Grado E.T.S.I. Informática : Universidad de Valladolid, 2012.
33. **Cruz, Gonzales.** karapanza.net. [En línea] [Citado el: 20 de Junio de 2015.] <http://www.karapanza.net/conectar-csharp-a-db2/>.
34. **JSC3D, Proyecto.** Visualizador de modelos web. [En línea] [Citado el: 16 de Junio de 2015.] <https://code.google.com/p/jsc3d/>.
35. **Blanco Alonso, Alvaro y Cardeñoso Payo, Valentín.** *Desarrollo centrado en el usuario de un juego "causa-efecto" sobre Android*. Trabajo Fin de Grado E.T.S.I. Informática : Universidad de Valladolid, 2014.
36. **Corrales Astorgano, Mario y Crespo, Yania.** *Asistencia a la movilidad de estudiantes por el campus universitario sobre Smartphone*. Proyecto Fin de Carrera E.T.S.I. Informática : Universidad de Valladolid, julio 2014.

37. Mundo table, librerías de realidad aumentada. [En línea] [Citado el: 20 de Junio de 2015.] <http://www.mundotablespain.com/Articulos/Programacion/APIs-realidad-aumentada/APIs-realidad-aumentada.htm>.

38. **Readthedocs**. Androidos.readthedocs.org. [En línea] [Citado el: 19 de Junio de 2015.] <http://androidos.readthedocs.org/en/latest/data/caracteristicas/>.

39. **SQLite**. SQLite. [En línea] [Citado el: 16 de Junio de 2015.] <https://www.sqlite.org/features.html>.

Anexo

A. Diagramas

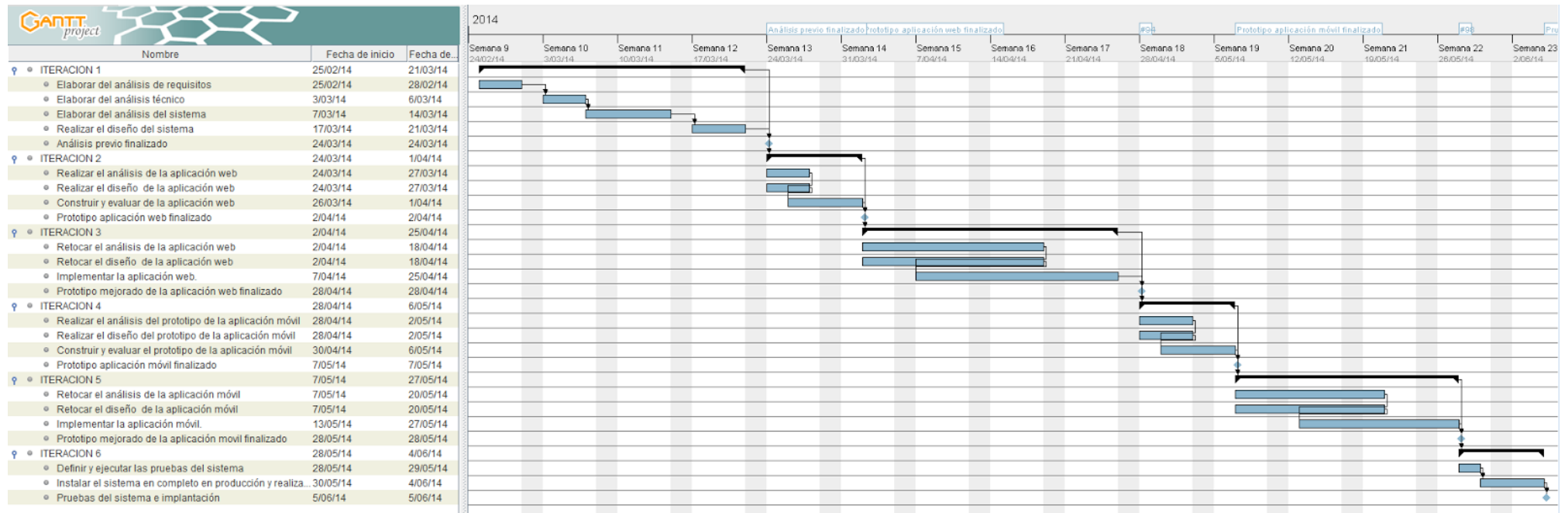


Ilustración 1 - Diagrama de Gantt del proyecto.

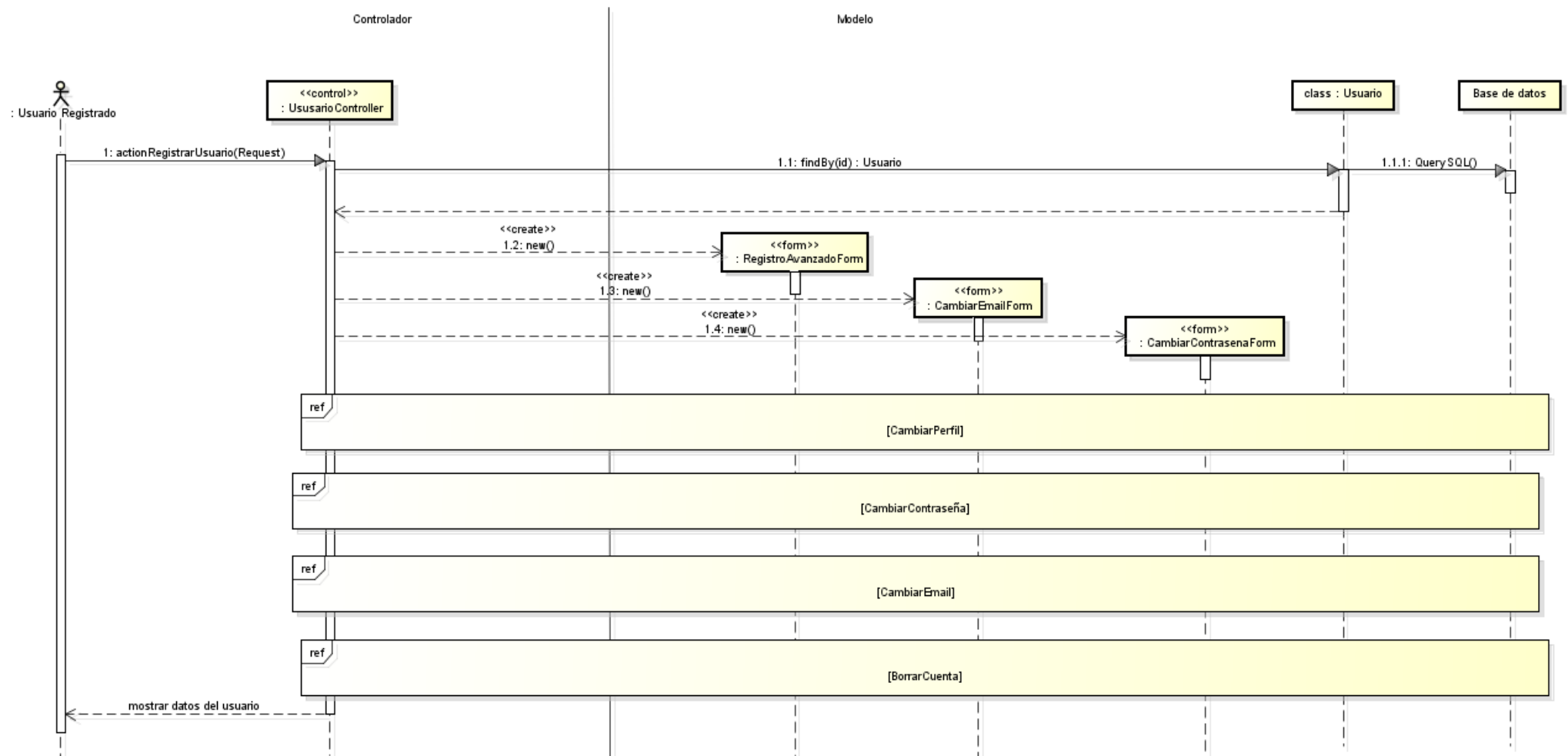


Ilustración 2 - Diagrama de secuencia de ver perfil de usuario.

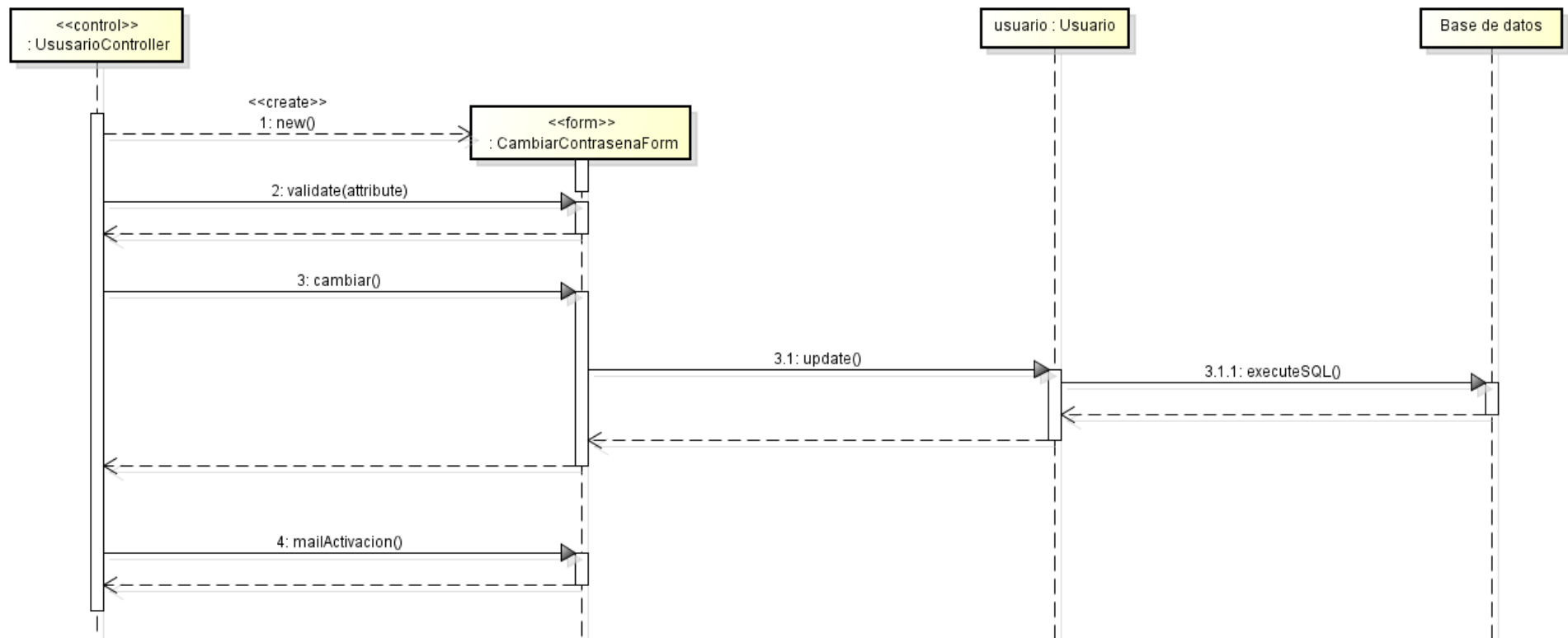


Ilustración 3 - Diagrama de secuencia de cambiar contraseña.

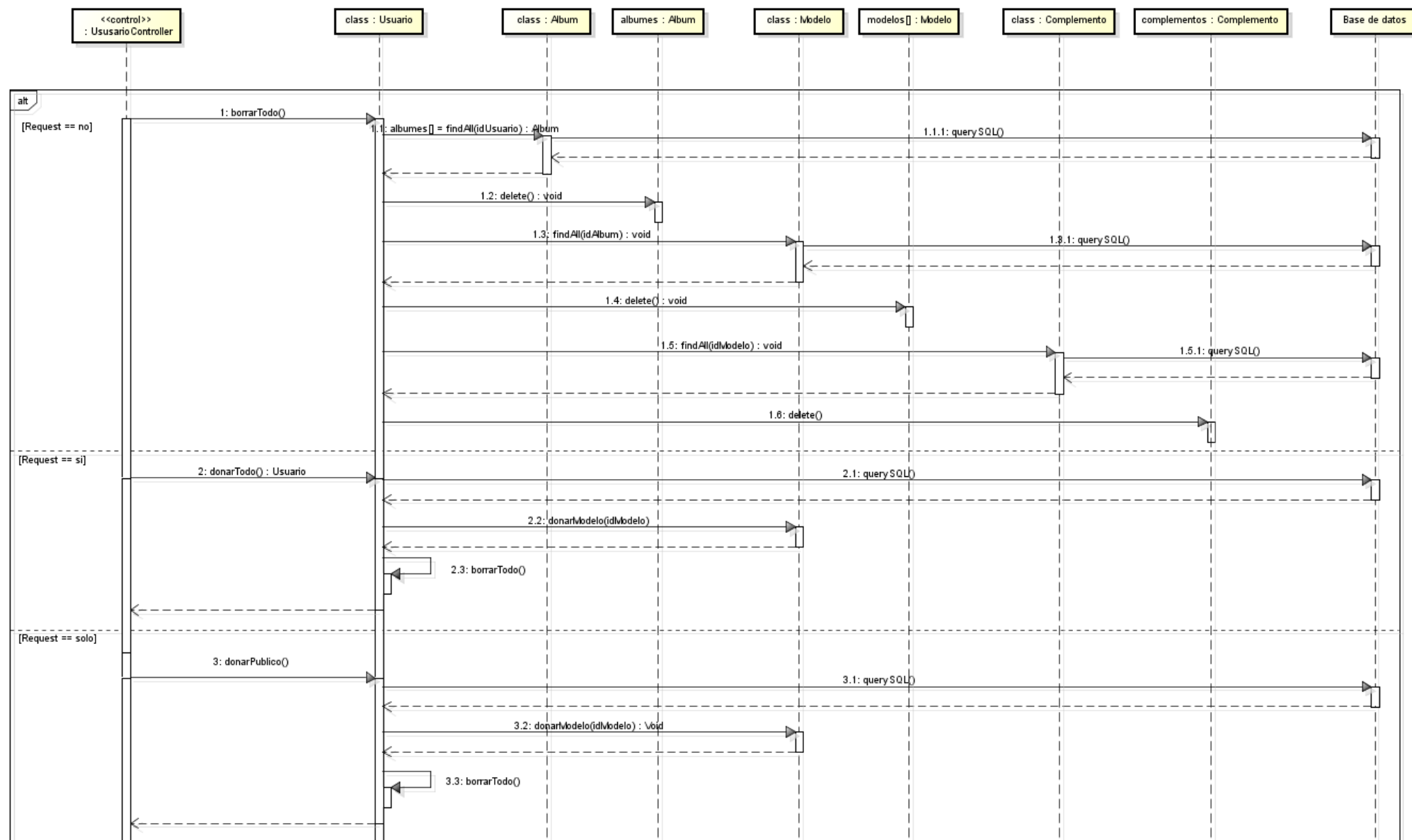


Ilustración 4 - Diagrama de secuencia de borrado de cuenta.

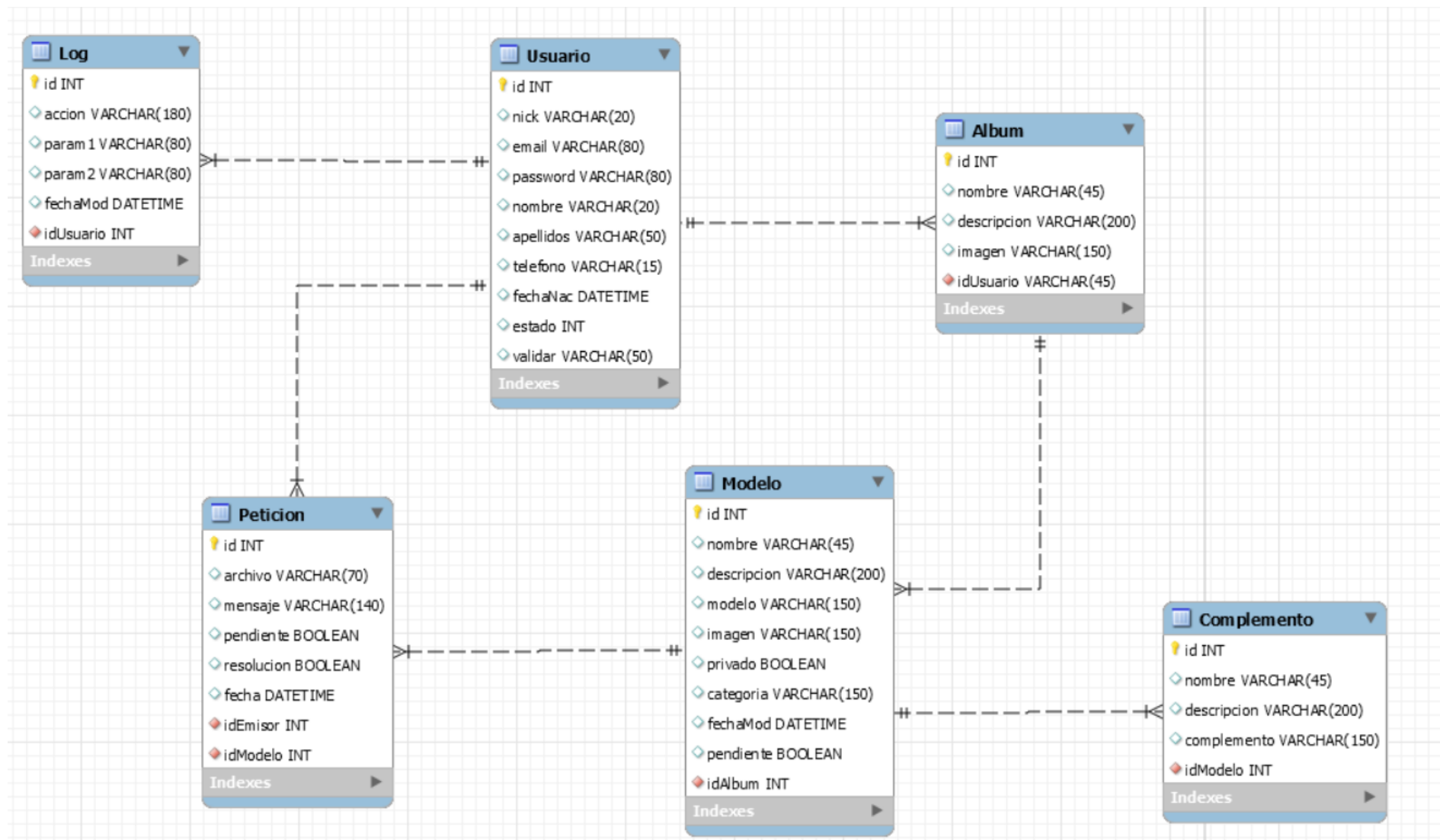


Ilustración 5 - Esquema de la base de datos de la aplicación web.

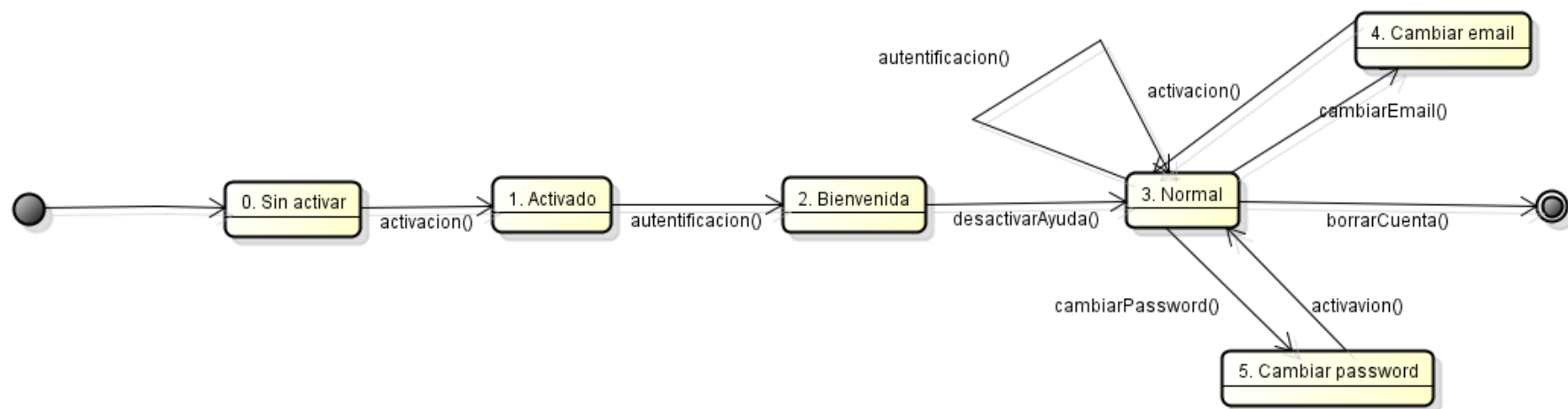


Ilustración 6 - Diagrama de secuencia de descargar modelo.

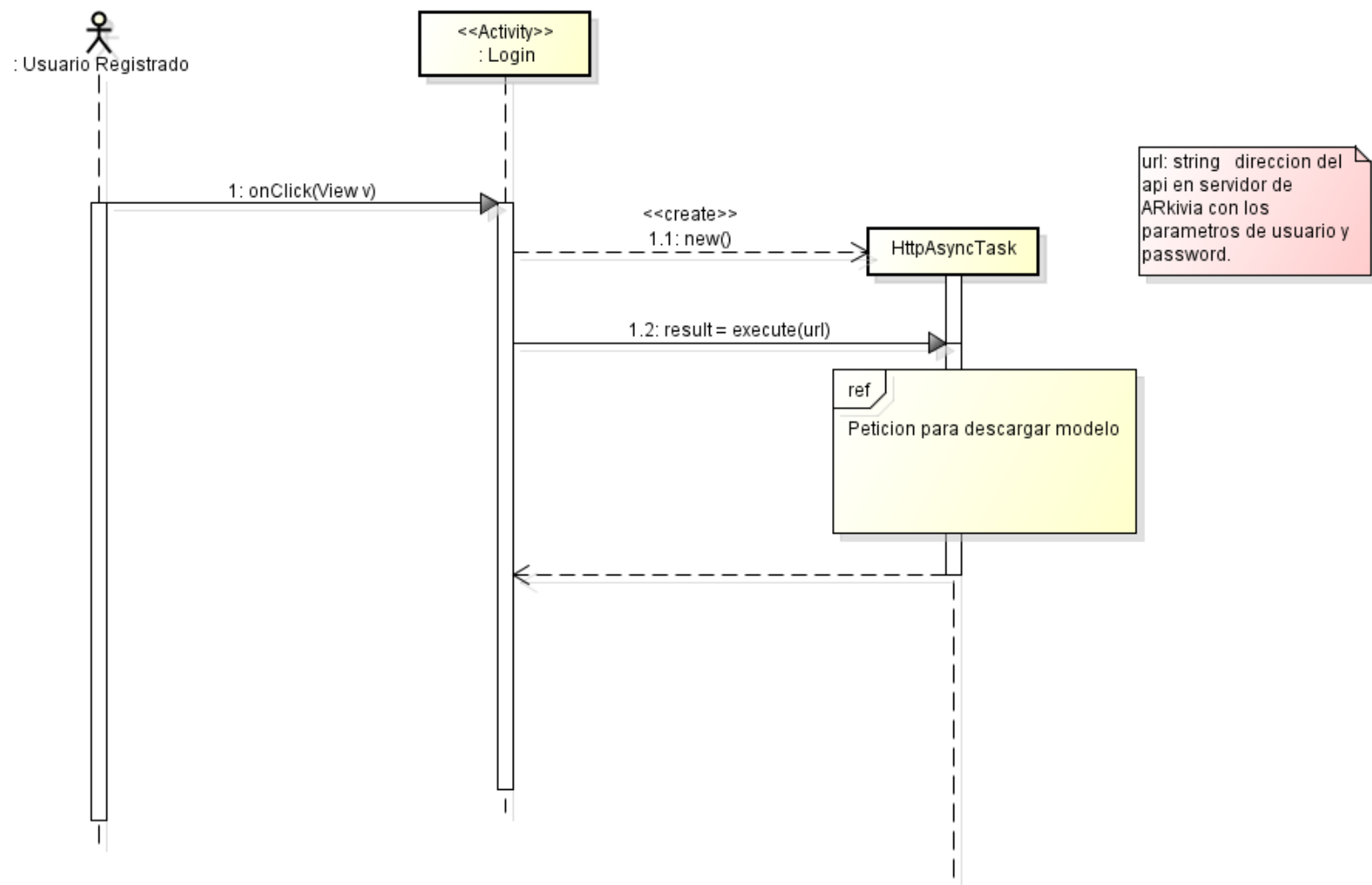


Ilustración 7 - Diagrama de secuencia de descargar modelo.

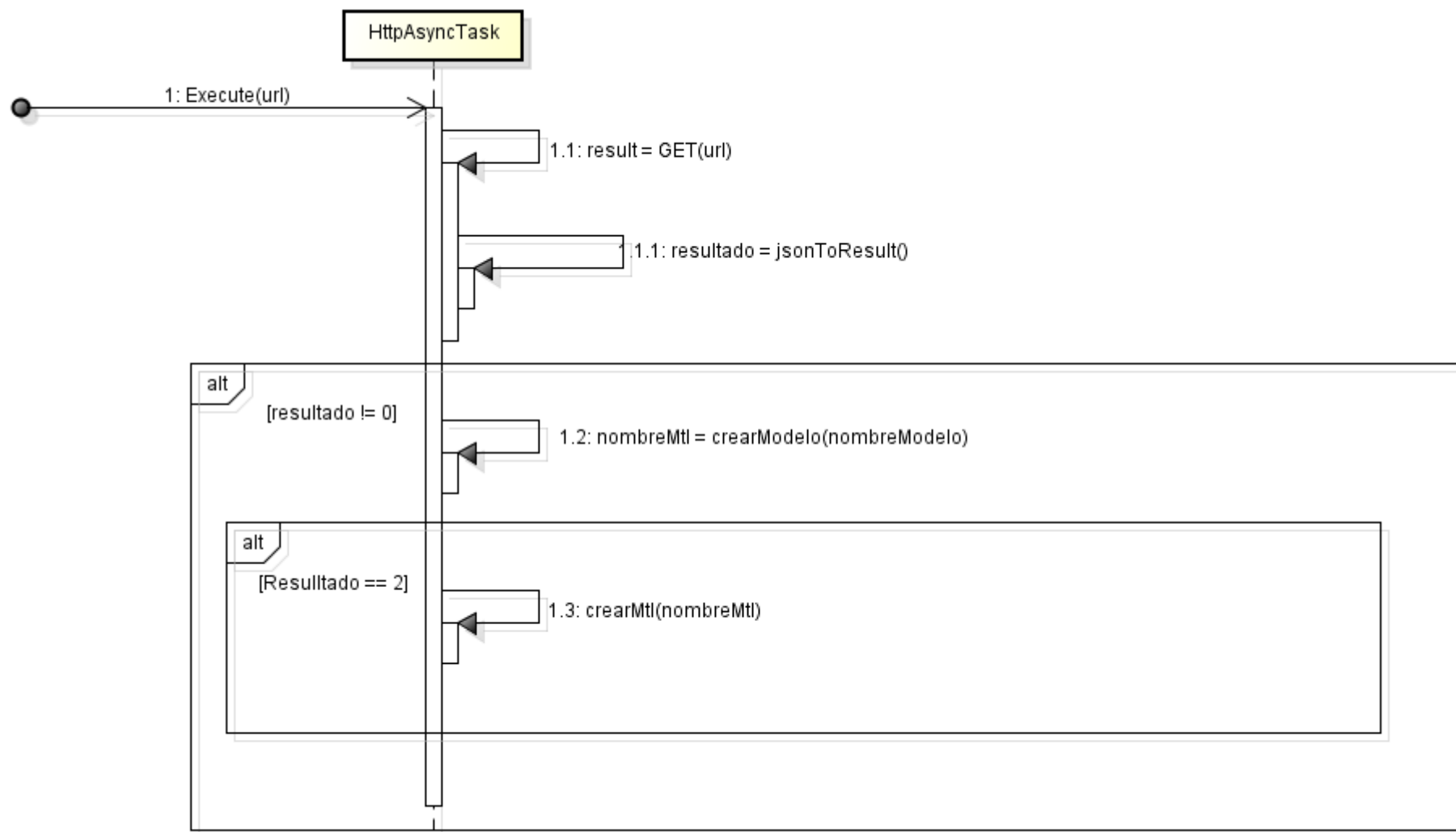


Ilustración 8 - Diagrama de secuencia de petición a API para descargar modelo.

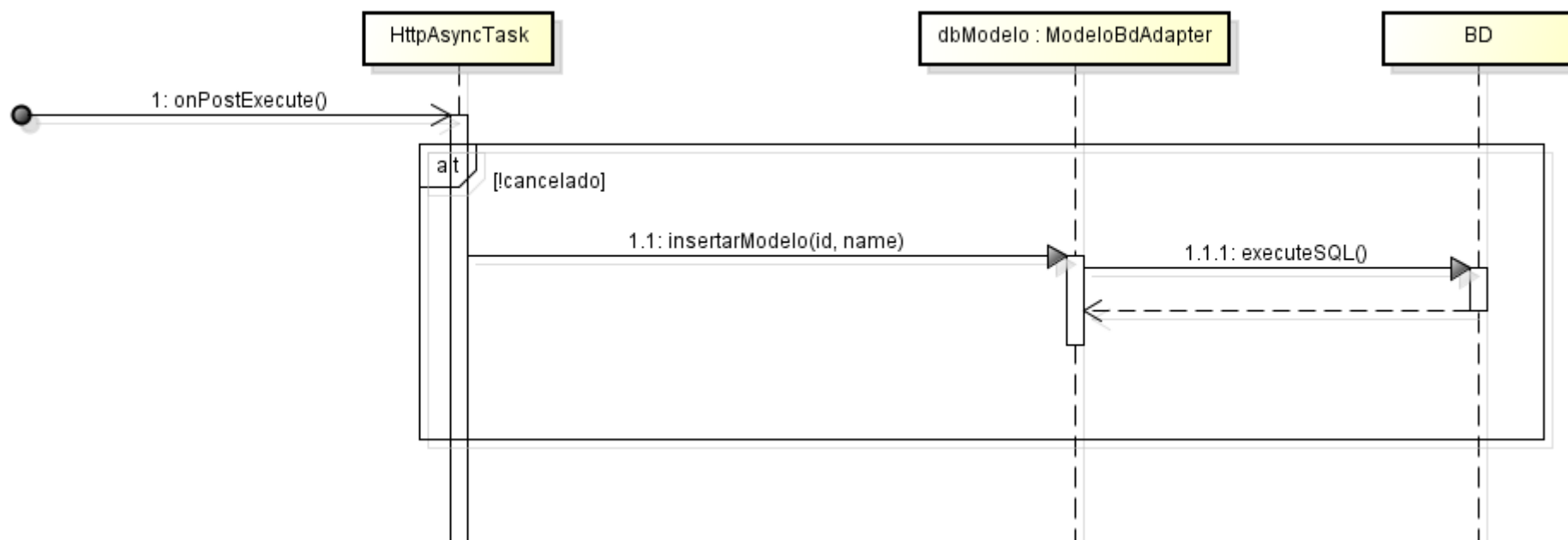


Ilustración 9 - Diagrama de secuencia de añadir modelo una vez creado el modelo.

B. Manuales de usuario

Las aplicaciones realizadas en este proyecto deben de tener unas instrucciones de buen uso para que los usuarios tenga un documento al que referirse en caso de dudas. Estos manuales están en un lenguaje muy simple, con muchas imágenes y con un índice resumido que nos permite que cualquier usuario pueda resolver sus problemas en un breve periodo de tiempo.

Se han hecho 2 manuales, uno por cada aplicación, que se encuentran en el CD-ROM, en la carpeta de Manuales

C. Manual de instalación

Como hemos mencionado a lo largo de la memoria, el proyecto debe llevarse a cabo hasta el punto de estar en producción y funcionamiento como los clientes finales desean. Es por esto que se ha realizado un manual de instalación que se encuentra en el CD-ROM, el cual se adjunta a este proyecto.

Tras realizar la instalación de las aplicaciones es aconsejable que se realice otra vez la batería de prueba que se define en esta memoria.

D. Contenido del CD-ROM

El contenido del CD-ROM que está junto al proyecto contiene la siguiente información:

- Un fichero memoria.pdf, con la documentación del proyecto.
- La carpeta *manuales*, que contiene los manuales de las aplicaciones y otro manual de como se instala la aplicación.
- La carpeta *código fuente*, que contiene toda la implementación relacionada con las dos aplicaciones, junto con el fichero de instalación de la app de ARkivia Viewer con el nombre ARkiviaViewer.apk.
- La carpeta *desarrollo*, que contiene el archivo ARkivia.asta donde se encuentran los diagramas desarrollados a lo largo del proyecto y el archivo "ARkivia diagrama de Gantt.gan" con el diagrama de Gantt.
- La carpeta *base de datos*, que donde se encuentra un archivo que contiene el código sql, el cual hay que ejecutar para crear las tablas en la base de datos de ARkivia.