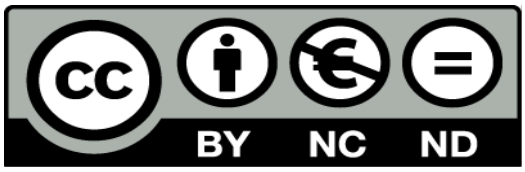
UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Aplicación móvil para el turismo en la ciudad de Logroño con tecnologías de realidad aumentada
Autor/es
Álvaro Pérez Sala
Director/es
Arturo Jaime Elizondo
Facultad
Facultad de Ciencias, Estudios Agroalimentarios e Informática
Titulación
Grado en Ingeniería Informática
Departamento
Curso Académico
2013-2014



publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad

Facultad de Ciencias, Estudios Agroalimentarios e Informática

Titulación

Grado en Ingeniería Informática

Título

APLICACIÓN MOVIL PARA EL TURISMO EN LA CIUDAD DE LOGROÑO CON TECNOLOGÍAS DE REALIDAD AUMENTADA

Autor/es

Álvaro Pérez Sala

Tutor/es

Arturo Jaime Elizondo

Departamento

Departamento de Matemáticas y Computación

Curso académico

2013 - 2014

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO

LOGRAR: APLICACIÓN MOVIL PARA EL TURISMO EN LA CIUDAD DE LOGROÑO CON TECNOLOGÍAS DE REALIDAD AUMENTADA

Autor: Alvaro Pérez Sala

Tutor: Arturo Jaime Elizondo

GRADO EN INGENIARÍA INFORMÁTICA

DEPARTAMENTO DE MATEÁTICAS Y COMPUTACIÓN

FACULTAD DE CIENCIAS ESTUDIOS AGROALIMENTARIOS E INFORMÁTICA

RESUMEN

A lo largo de 4 meses hemos trabajado para desarrollar una aplicación para dispositivos móviles que hemos denominado LogrAR. Esta aplicación consiste en una guia turística con funcionalidades sociales, mediante la que los usuarios pueden visitar los monumentos y lugares de interés solo con su smartphone.

La característica a destacar de esta aplicación es el uso de realidad aumentada para mostrar información al usuario. La realidad aumentada es un nuevo paradigma de interacción usuario-ordenador que consiste en la inclusión de elementos virtuales dentro del mundo real. Es decir llevado a nuestra aplicación mostrar sobre los lugares de interés, monumentos, etc información, objetos o contenido multimedia que nos permitan conocer la historia de cada uno de esos puntos turísticos y asi comprender su importancia.

La parte de red social de la aplicación ha requerido de la implementación de una aplicación de servicios donde se centraliza toda la gestión y administración de sus características sociales. Se ha optado por el uso de una serie de servicios web implementados con ASP.NET, junto con toda la implementación de gestión subyacente desarrollada en .NET trabajando sobre una base de datos SQL Server. Todo se ha desplegado en un servidor virtual facilitado por la universidad.

Para el desarrollo de la aplicación móvil se ha usado el entorno de desarrollo Unity, este fue elegido por las facilidades que nos da para implementar contenidos en realidad aumentada, aunque prsente otras carencias. La tecnología de programación usada en este entorno es .NET Framework y sus proyectos son compilables tanto para iOS como para Android. Como inconveniente ha sido necesaria la implementación de un API que nos permita la gestión sencilla de interfaces de usuario en 2D. Este API nos automatiza la gestión de eventos, la navegación entre vistas, asi como el comportamiento de los distintos controles en la pantalla.

La parte de realidad aumentada se ha implementado con el SDK de de Qualcomm “Vuforia”, y dada la limitación dl proyecto se ha enfocado en solo tres puntos de la ciudad.

La aplicación ha sido desarrollada para dispositivos iOS por lo que la versión final de la aplicación es para esta plataforma. Aunque también se ha generado una adaptación para Android, el funcionamiento es limitado, ya que la optimización del software se ha hecho para iOS. Por ello proponemos como mejora una futura optimización para Android fuera de este proyecto.

Por último podemos ver una muestra de los resultados obtenidos en el siguiente video:
http://youtu.be/XmM2O3clbQs

ABSTRACT

Along 4 mounths we've been working in develop a mobile application that we've called "LogrAR". This application consists in a turistic guide with social features by mean of it users can visit monuments, and interesting places only with their smartphone.

The main feature of this application is the that it use augmented reallity to show information to the user.Augmented reallity is a new paradigm of interaction between user and computer. It consist in include virtual elements into the real world. In our application it shows information, objects, or media contents which allow us knew the history of the monuments and places and show us the importance of them.

The social part of the apliction has need the development of a services application where we have centralizad all the management of its social features. For this we have choosen web services developed in ASP.NET whith a management application in .NET and all of this working with a SQL Server database. These services has been deployed in a server provided by University of La Rioja.

For the development of the mobile application we have used Unity as integrated development environment, this was selected for being easy for implement augmented reallity contents even though it has any lacks. The programming technology that we have used in this environment has been .NET framework, and its projects can be built for iOS and Android platforms. As an issue we have needed to develop an API that give us tools for an easier management of graphic user interfaces in Unity. This API make easier working with events, the nevegability between different views, and the behaviour of the controls in the screen.

Augmented reallity contents have been developed with the Qualcomm SDK "Vuforia", and taking the time limitations into account we have only devloped three points of the city.

The aplication has been developed for iOS devices, however the final version it's availabloe for this platform. But also an Android adaptation has been generated, it has limited features, because the software optimization has been done for iOS. By that we suggest as improvement a future optimization for Android out of this project.

Finally we can see a sampe of our results in this video:

http://youtu.be/XmM2O3clbQs

CONTENIDO

RESUMEN 1

ABSTRACT 2

INTRODUCCIÓN 5

MOTIVACIONES 5
OBJETIVO 5

DESARROLLO DEL PROYECTO 6

ANÁLISIS DE REQUISITOS 7
1. ANÁLISIS DE DATOS 7
2. ANÁLISIS DE REQUISITOS FUNCIONALES 8
3. ANÁLISIS DE REQUISITOS NO FUNCIONALES 8

ALCANCE DEL PROYECTO 10
PLANIFICACIÓN 13
1. HITOS 13
2. CRONOGRAMA Y DESGLOSE DE TAREAS 13

DISEÑO 18
1. DISEÑO DE LA BASE DE DATOS 18
2. DISEÑO DE GESTIÓN DE APLICACIÓN 19
3. DISEÑO DE LA APLICACIÓN MÓVIL 21

IMPLEMENTACIÓN 28
1. IMPLEMENTACIÓN DE GESTIÓN DE APLICACIÓN 28
2. IMPLEMENTACIÓN DE LA APLICACIÓN MÓVIL 30

DESPLIEGUE Y PRUEBAS DEL SISTEMA 34
1. APLICACIÓN DE SERVICIOS WEB (PARTE SERVIDOR) 34
2. APLICACIÓN MÓVIL (PARTE CLIENTE) 36

SEGUIMIENTO Y CONTROL 38
1. SEGUIMIENTO DE TAREAS 38
2. CONTROL DE CAMBIOS EN LA PLANIFICACIÓN 40

CONCLUSIONES 43

RESULTADO FINAL **44**
ASPECTOS LEGALES **45**
1. LEY ORGÁNICA DE PROTECCIÓN DE DATOS 45
2. LICENCIA DE DISTRIBUCIÓN 45
MEJORAS **46**
LECCIONES APRENDIDAS **47**

BIBLIOGRAFIA **48**

AGRADECIMIENTOS **49**

INTRODUCCIÓN

Con motivo de la asignatura de "Trabajo fin de Grado" del Grado de Ingeniaría en Informática por la Universidad de la Rioja. Se presenta el siguiente proyecto para el desarrollo de una aplicación informática móvil usando tecnologías de realidad aumentada.

MOTIVACIONES

A lo largo del periodo de prácticas hemos estado trabajando e investigando con tecnologías para el desarrollo e implementación de interfaces en realidad aumentada. Apostando por estas tecnologías nuevas e innovadoras que cambian la forma en el que el usuario interacciona con los sistemas de información se plantea su aplicación al sector del turismo en la ciudad de Logroño. Así aprovecharemos las nuevas herramientas que nos proporciona esta tecnología con el uso de plataformas comunes de las que todos disponemos (teléfonos móviles, smartphone).

OBJETIVO

El objetivo del proyecto será por tanto el desarrollo de una aplicación para la plataforma móvil IOS para el fomento y desarrollo del turismo en la ciudad de Logroño.

Dicha aplicación integrará contenidos en realidad aumentada, para la visualización de diferentes monumentos, estatuas, lugares emblemáticos, y demás puntos de interés turístico junto con información sobre ellos, enlaces de interés, y otros contenidos. Esta información se mostrará de manera dinámica e interactiva para el usuario, será lo más intuitivo posible el acceso y uso de esta. También para la exposición de esta información se usarán tecnologías TTS (Text To Speech), de forma que el usuario además de ver información en interfaces de realidad aumentada, también podrá recibir una explicación hablada del mismo.

A estas nuevas interfaces de realidad aumentada le añadiremos funcionalidades sociales, de manera que cada usuario registrado en el sistema puede compartir información con el resto, o con los usuarios a los que esté conectado. También podemos compartir rutas turísticas en la ciudad, fotos en los sitios de interés y opiniones personales.

Los usuarios podrán ver información compartida por sus contactos, o por otras personas sobre cada sitio visitado. La gestión del sistema y el almacenamiento de contenidos se hará en la nube, mediante el despliegue de un servidor que centralice el funcionamiento de la aplicación.

Se pretende diseñar el sistema de manera que tras la finalización del proyecto se pueda desarrollar una interfaz web que nos muestre el contenido utilizado en el teléfono, lugares visitados o fotos realizadas desde la aplicación.

DESARROLLO DEL PROYECTO

ANÁLISIS DE REQUISITOS

1. ANÁLISIS DE DATOS

El sistema almacenará usuarios, de cada usuario guardaremos nombre, apellidos, login o nick que debe de ser único para cada usuario, lugar de nacimiento, edad, sexo y correo electrónico. Además de una forma adecuada para proteger los datos se guardará el password encriptado de acceso de cada uno de los usuarios.

Para cada uno de los lugares o sitios de interés almacenaremos un número de serie para identificar el material multimedia asociado, el cual debe de ser único. Junto a este almacenaremos un título para el lugar. Un lugar tendrá su posición GPS formada por latitud y longitud, no pudiendo haber dos lugares con la misma posición

Cada lugar podrá ser visitado por los usuarios de la aplicación, en el momento que accedan a la aplicación en uno de los lugares definidos se registrará una visita a ese lugar, pudiendo cada usuario visitar varios sitios y un sitio ser visitado por varios usuarios varias veces. Sería interesante conocer el número de veces que un usuario ha visitado cada lugar.

Cada usuario también podrá contactar con otros usuarios y estar en contacto con ellos. Se guardará la fecha en la que contactaron. Así podrán ver los sitios que han visitado, etc.

Al visitar un lugar, un usuario puede comentar el lugar o dar su opinión, pudiendo ser el comentario un texto con un máximo de 140 caracteres, y con una fecha de publicación. Se permitirá subir fotos que haya realizado con la aplicación para que las puedan ver los demás usuarios o solo la gente con la que está conectado. De cada foto guardaremos el nombre del archivo correspondiente junto con el usuario que la ha subido, el lugar y la fecha en el que lo ha hecho, almacenaremos si el usuario la ha marcado como pública o en su defecto es privada. También se podrán comentar las fotos, identificando el usuario que ha comentado cada foto y la fecha en la que lo hizo.

Cuando un usuario solicite una baja pasará a estar registrado como usuario inactivo, almacenando también así la fecha de baja. Todo el material correspondiente a un usuario antiguo no será accesible por ningún otro. En un plazo de tres meses podrá solicitar una rehabilitación de la cuenta, por lo que esta volverá a ser visible en el mismo estado anterior. Quedará constancia en el sistema la baja realizada con fecha de inicio y fin, aquel usuario con fecha de baja pero sin fecha de vuelta se considerará usuario inactivo.

2. ANÁLISIS DE REQUISITOS FUNCIONALES

Se desarrollará un sistema para la visita turística a Logroño, por lo que permitirá a los usuarios entre otras, visitar la ciudad viendo contenido turístico (imágenes, explicaciones, videos…) en realidad aumentada. En la Fig. 1 encontramos el análisis de la funcionalidad que recogerá la aplicación. Se omite la descripción detallada de cada caso de uso, ya que el mismo título nos da la información suficiente para comprender en qué consiste cada uno de ellos. Los casos de uso incluidos en rojo serán limitados, es decir con el fin de ajustarnos a la duración estimada del proyecto, no serán incluidos en la interfaz del cliente móvil pero si en el resto del sistema.

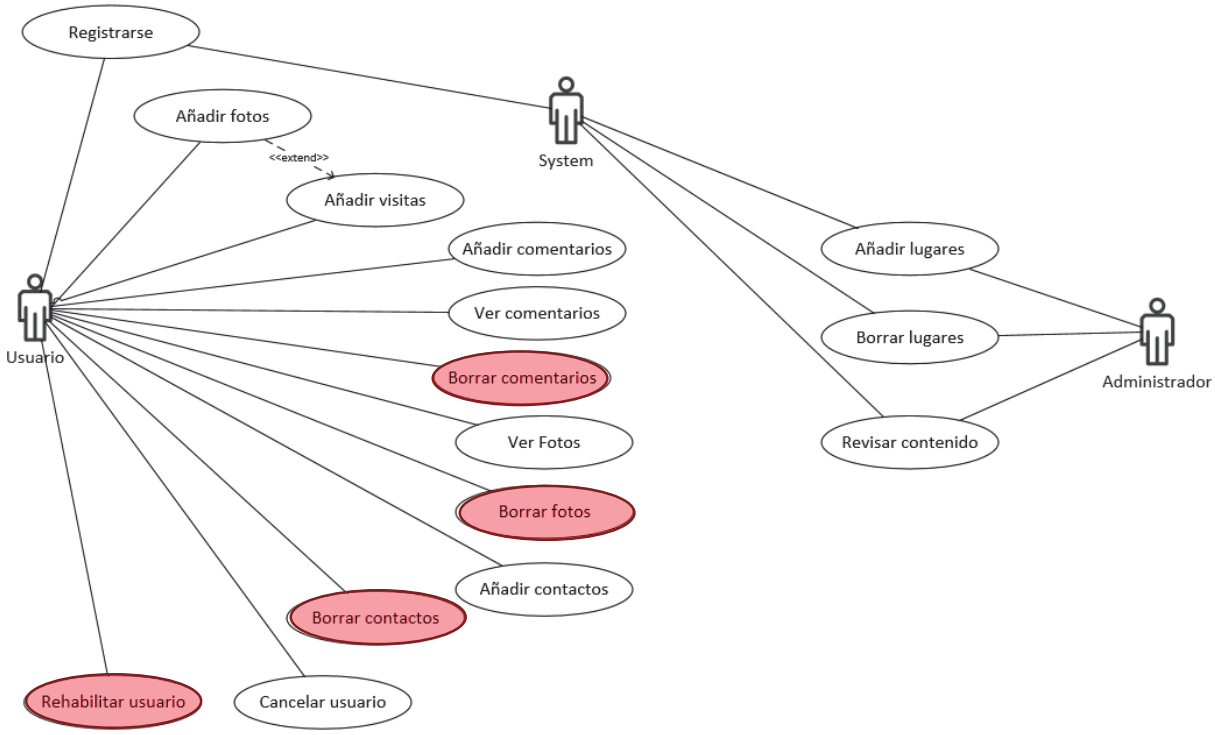


Fig. 1: Diagrama de casos de uso del sistema.

3. ANÁLISIS DE REQUISITOS NO FUNCIONALES

La aplicación además de su parte visual requiere un sistema de gestión, que se realizará mediante un servidor en línea que además de administrar el contenido dinámico de la aplicación se encargará de gestionar la información que esta use para su funcionamiento.

Los usuarios, información sobre lugares, comentarios, visitas, recursos relacionados etc. Se almacenarán en una base de datos centralizada, diseñada y optimizada para tal fin.

Para servir el contenido multimedia a la aplicación y almacenar el generado por los usuarios usaremos también una aplicación de gestión para servir el contenido adecuado. Esta quedará integrada en el mismo servidor.

La aplicación móvil actuará como cliente y solicitará al servidor todo el contenido que requiera para su uso. Será imprescindible por tanto el disponer de conexión permanente a internet para el uso de la misma. Esta no tendrá ni información ni lógica para el manejo de esta, todo ello se delegará en el servidor. Por tanto solo será presentación de la interfaz de usuario y del contenido multimedia. La aplicación será compatible con las plataformas iOS e Android, y poder más tarde ser ampliable a otras plataformas.

ALCANCE DEL PROYECTO

Vamos a definir el alcance del proyecto según la estructura de descomposición de trabajo que podemos ver en la Fig. 2 y detallada a continuación.

1.01.01. Despliegue del entorno: Instalación de sistema Windows Server 2008, IIS 7, SQL Server 2012 y Web Deploy en el host servidor, configuración de estas utilidades y acceso público a la red.

1.01.02. Desarrollo de BD: Diseño, implementación y despliegue en SQL Server 2012 de la base de datos a partir del análisis de datos proporcionado.

1.01.02.1. Gestión BD: Diseño orientado a objetos e implementación del módulo de conexión y acceso a la base de datos. Usando la tecnología ADO de .NET Framework.

1.01.02.2. Gestión multimedia: Implementación de la lectura de archivos y generación de enlaces web para la descarga de contenido multimedia, y combinarlo con la información tomada de la BD.

1.01.03. Implementación de servicios: Diseño e implementación de la lógica de aplicación y el acceso a esta mediante servicios web públicos. Empleando las tecnologías de ASP.NET para servicios web SOAP por lo que podría ser necesaria una pequeña investigación sobre ellos.

1.01.04. Pruebas de funcionamiento: Pruebas de funcionamiento de cada uno de los métodos proporcionados por los servicios web

1.02.01. Diseño: Diseño gráfico de la interfaz de la aplicación móvil, así como la confección de dibujos y texturas para esta si fuera necesario, presentación en bocetos de las vistas de la aplicación.

1.02.02. Estudio, material e información: Estudio de los puntos de interés turístico de Logroño, recogida de fotos de cada uno de ellos y búsqueda de información relacionada. Para ajustarnos a la duración y límites del proyecto, se restringe este paquete de trabajo a los 3 puntos más relevantes de la ciudad.

1.02.03. Implementación interfaz usuario: Para la implementación de la app móvil se usará el entorno Unity3D, se implementarán en éste paquete de trabajo el acceso a los servicios junto con la interfaz de usuario. Los casos de uso Borrar comentario, foto y usuario, así como la rehabilitación de estos no se va a incluir en la primera versión por salirse de los límites del proyecto. Pero el diseño permitirá su fácil inclusión en futuras versiones.

Para el desarrollo de la interfaz de usuario, se va a diseñar e implementar una librería auxiliar que permita el desarrollo de alto nivel de interfaces gráficas en 2D, todo ello usando .NET Framework 2.0 con el que trabaja Unity.

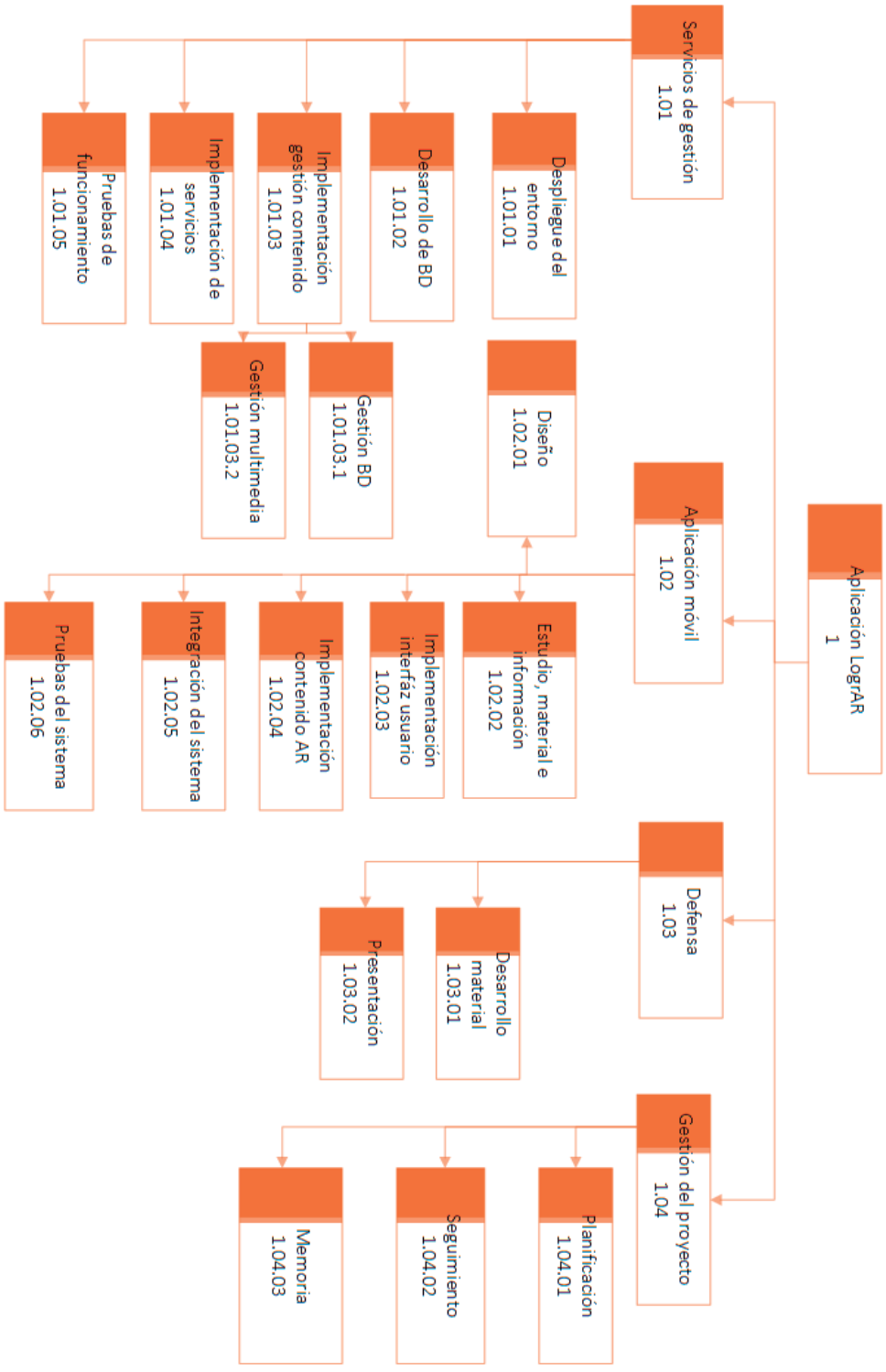

Aplicación LogrAR
1
Servicios de gestión
1.01
Despliegue del entorno
1.01.01
Desarrollo de BD
1.01.02
Implementación gestión contenido
1.01.03
Implementación de servicios
1.01.04
Pruebas de funcionamiento
1.01.05
Gestión BD
1.01.03.1
Gestión multimedia
1.01.03.2
Aplicación móvil
1.02
Diseño
1.02.01
Estudio, material e información
1.02.02
Implementación interfaz usuario
1.02.03
Implementación contenido AR
1.02.04
Integración del sistema
1.02.05
Pruebas del sistema
1.02.06
Defensa
1.03
Desarrollo material
1.03.01
Presentación
1.03.02
Gestión del proyecto
1.04
Planificación
1.04.01
Seguimiento
1.04.02
Memoria
1.04.03

Fig. 2: EDT

1.02.04. Implementación contenido AR: Desarrollo de las escenas de realidad aumentada de la aplicación, se realizarán tres escenas en Unity3D con diferente tipo de contenido, coincidiendo con cada uno de los tres sitios definidos en el apartado de investigación y recogida de información.

1.02.05. Integración del sistema: Integración de la interfaz plana de usuario con las escenas de realidad aumentada, seleccionando por posicionamiento dinámicamente la escena que corresponda en cada momento.

1.02.06. Pruebas del sistema: Pruebas del sistema completo y test de la aplicación, corrección de los errores más importantes dado el caso.

1.03.01. Desarrollo material: Desarrollo de la presentación y preparación de la exposición de la defensa. Grabación de un breve video para la presentación de la aplicación.

1.03.02. Presentación: Presentación del trabajo ante el tribunal correspondiente con una exposición de límite 15 minutos.

1.04.01. Planificación: Elaboración de la planificación temporal del proyecto, desglose de tareas y dedicación a cada una de ellas.

1.04.02. Seguimiento: Control del desarrollo del proyecto, revisión semanal de horas invertidas comparadas con las planificadas. Elaboración de posibles re planificaciones si fueran necesarias.

1.04.03. Memoria: Desarrollo de una memoria del trabajo según los estándares indicados por la Universidad de La Rioja incluyendo toda la información sobre el proyecto realizado.

PLANIFICACIÓN

Para la realización del proyecto se dispone de un máximo de 300 horas repartidas en 285 para trabajo y 15 horas para el seguimiento y control del proyecto con el tutor asignado. Además 10 de las horas de trabajo serán empleadas para la preparación de la defensa, y 5 a la planificación del proyecto. Para el la parte de desarrollo del servidor emplearemos 125 de las 285 horas, y para el cliente móvil 145 horas de trabajo dado que tiene una mayor carga que lo anterior.

El proyecto se desarrollara en 23 semanas distribuidas en 15 semanas de desarrollo (1-15), dos semanas quedan ociosas a fines de poder dedicarlas al estudio de los exámenes del segundo cuatrimestre (16 y 17), tres más para la revisión del proyecto (18 - 20), y las tres últimas (20-23) para la preparación y defensa del proyecto.

1. HITOS

Se definen los principales hitos del proyecto como:

- **Presentación planificación –** Semana 01 (3-7 Febrero)
- **Despliegue del servidor –** Semana 07 (17-21 Marzo)
- **Recopilación material –** Semana 09 (31 M – 4 Abril)
- **Fin de app móvil –** Semana 16 (19 – 23 Mayo)
- **Entrega del proyecto –** Semana 21 (23 – 27 Junio)
- **Defensa del proyecto –** Semana 24 (14 – 16 Julio)

La semana 12 (21 de Abril) se salta por ser festiva.

Podemos ver el diagrama de hitos en Fig. 3.

2. CRONOGRAMA Y DESGLOSE DE TAREAS

Los paquetes de trabajo se organizarán en tareas según se desglosa a continuación. Podemos encontrar el cronograma correspondiente en las figuras: Fig. 4 y Fig. 5.

DESARROLLO DE GESTIÓN DE APLICACIÓN (EDT 1.01)

- **Despliegue del entorno en el servidor** (EDT 1.01.01) – Se asignan 4 horas de trabajo a realizar entre 4 y 5 de Febrero
- **Diseño e implementación de la base de datos** (EDT 1.01.02) – 6 horas de trabajo repartidas entre 5 y 7 de Febrero
- **Implementación parte de gestión de BD** (EDT 1.01.03.1) – 12 horas de trabajo total repartidas entre 10 y 12 de Febrero

Hitos \ Semana	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Presentación planificación	◆																							
Despliegue del servidor							◆																	
Recopilación material									◆															
Fin app móvil																◆								
Presentación del TFG																					◆			
Defensa del TFG																								◆

Fig. 3: Diagrama de Hitos

- **Implementación parte de gestión de contenido** (EDT 1.01.03.2) – 28 horas de trabajo asignado, a realizar entre 13 y 24 de Febrero
- **Implementación parte de lógica y servicios** (EDT 1.01.04) – 65 horas de trabajo para investigación y desarrollo del paquete de trabajo a realizar entre 25 de Febrero y 20 de Marzo
- **Pruebas de funcionamiento del servidor** (EDT 1.01.05) – 4 horas a realizar el 21 de Marzo

DESARROLLO APLICACIÓN MÓVIL (EDT 1.02)

- **Diseño de la interfaz y confección de material necesario** (EDT 1.02.01) – Designadas 10 horas para esta tarea, a completar entre 24 de Marzo y 14 de Abril
- **Recogida de material e información del sitio de interés seleccionado** (EDT 1.02.02) – 8 horas de trabajo entre 28 de Marzo y 1 de Abril
- **Implementación de la interfaz de usuario** (EDT 1.02.03) – 48 horas de trabajo entre el 2 de Abril y el 14 de Mayo
-
- **Implementación de realidad aumentada** (EDT 1.02.04) – 40 horas de trabajo entre el 29 de Abril y el 14 de Mayo
- **Integración de todo el sistema** (EDT 1.02.05) – 14 horas de trabajo entre el 15 y 23 de Mayo.
- **Pruebas del sistema completo** (EDT 1.02.06) – 13 horas de trabajo entre el 21 y 23 de Mayo

DEFENSA DEL PROYECTO (EDT 1.03)

- **Desarrollo de material para la defensa del proyecto** (EDT 1.03.01 + 1.03.02) – Asignadas 10 horas para la preparación de la defensa entre 25 de Junio y 16 de Julio

GESTIÓN DEL PROYECTO (EDT 1.04)

- **Planificación del proyecto** (EDT 1.04.01) – 5 horas los días 3 y 4 de Febrero
- **Memoria, seguimiento y control del proyecto** (EDT 1.04.02 + 1.04.03) – 15 horas de trabajo a lo largo de todo el proyecto.

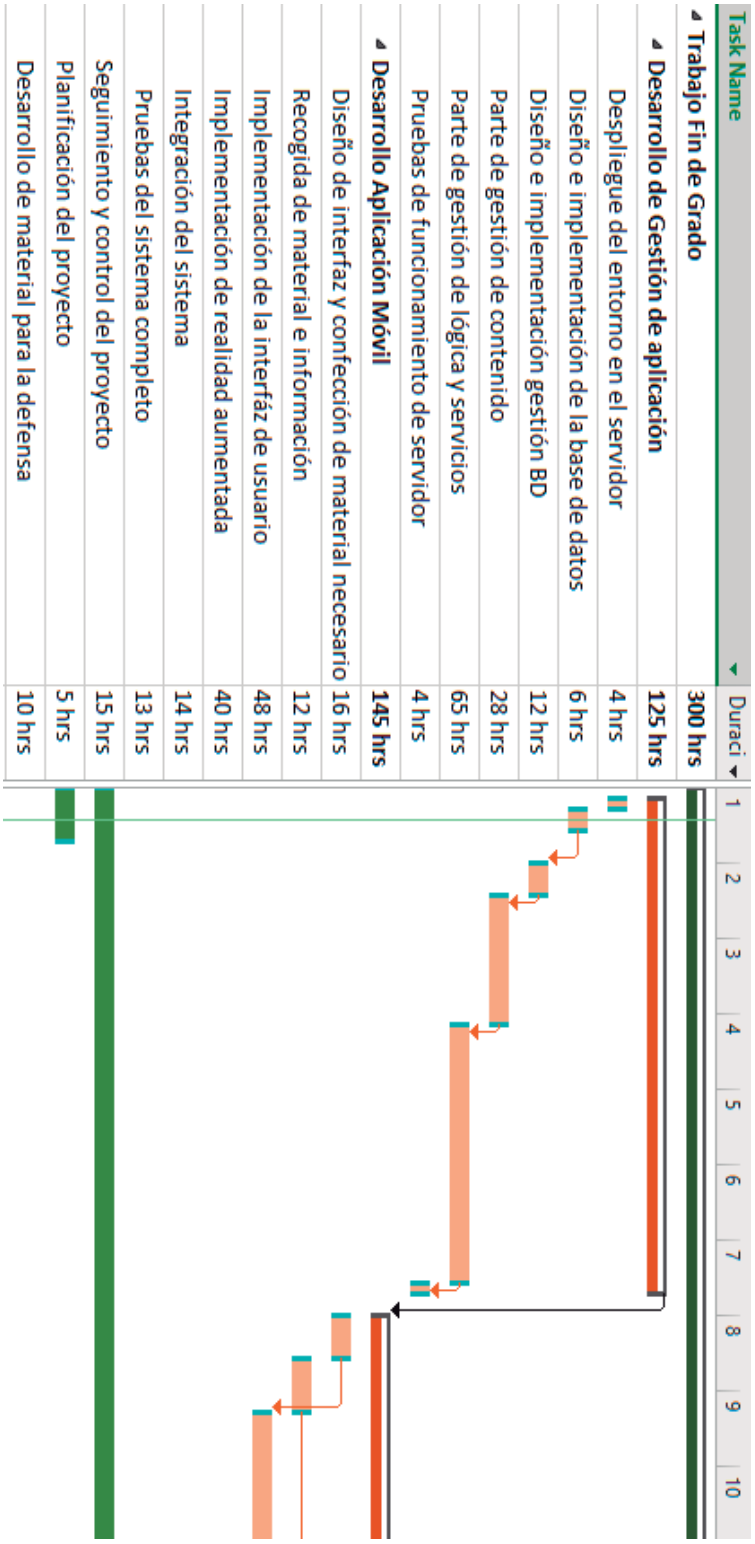


Fig. 4: Diagrama de Gannt 1/2 (Por semanas de trabajo)

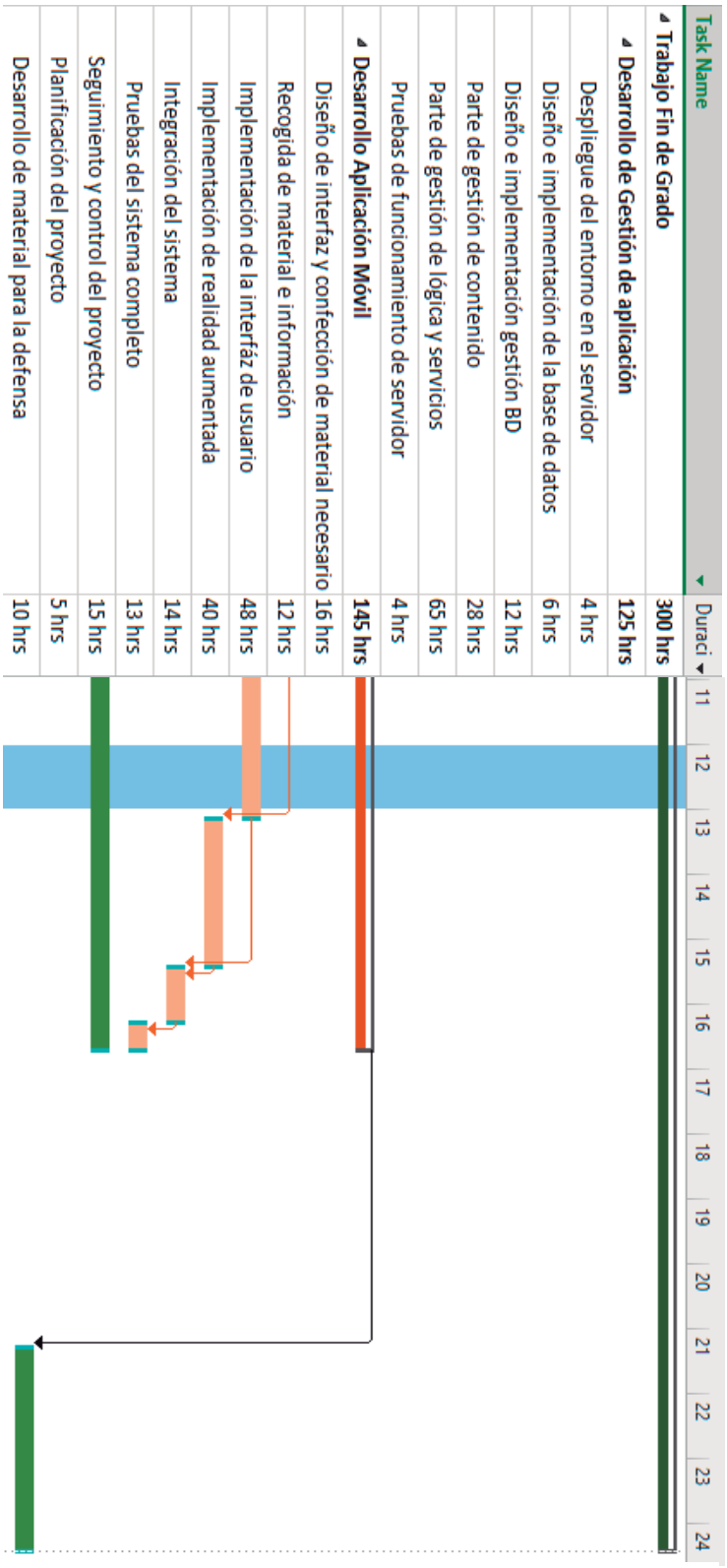


Fig. 5: Diagrama de Gannt 2/2 (Por semanas de trabajo)

DISEÑO

Como hemos visto anteriormente el sistema estará compuesto por dos partes separadas. Una de ellas alojada en un servidor, y la segunda en un cliente de un dispositivo móvil. Por tanto diseñamos este de manera distribuida con un solo servidor para acceder a la gestión y varios clientes que conecten con este. Podemos ver un esquema de la arquitectura de aplicación en la Fig. 6.

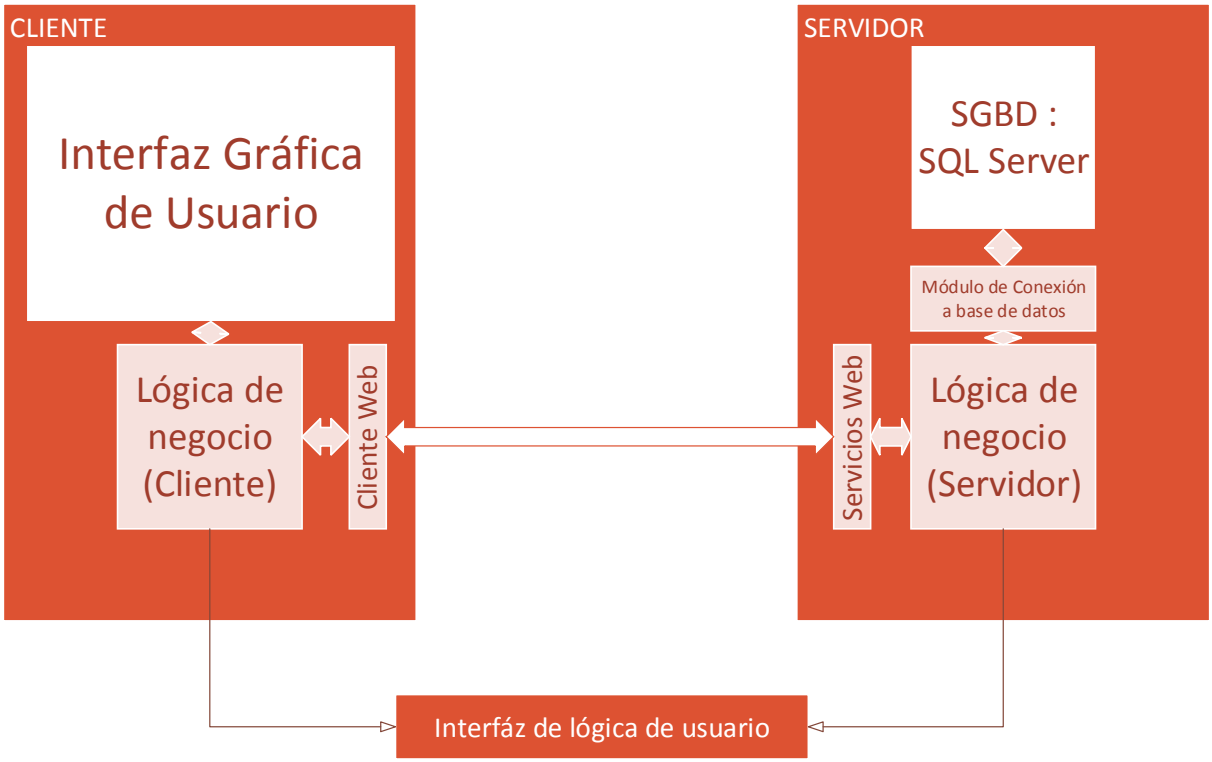


Fig. 6: Diagrama de arquitectura del sistema.

1. DISEÑO DE LA BASE DE DATOS

A partir del análisis de datos proporcionado, e incluido en el apartado Análisis de datos, diseñamos la base de datos como se refleja en la Fig. 7, tras ello desarrollamos el modelo relacional para la BD y orientado a objetos de cara a la aplicación. Podemos encontrar ambos diagramas en **Anexo I: Diseño - APS.LogrAR.DataModel, Modelo Relacional**

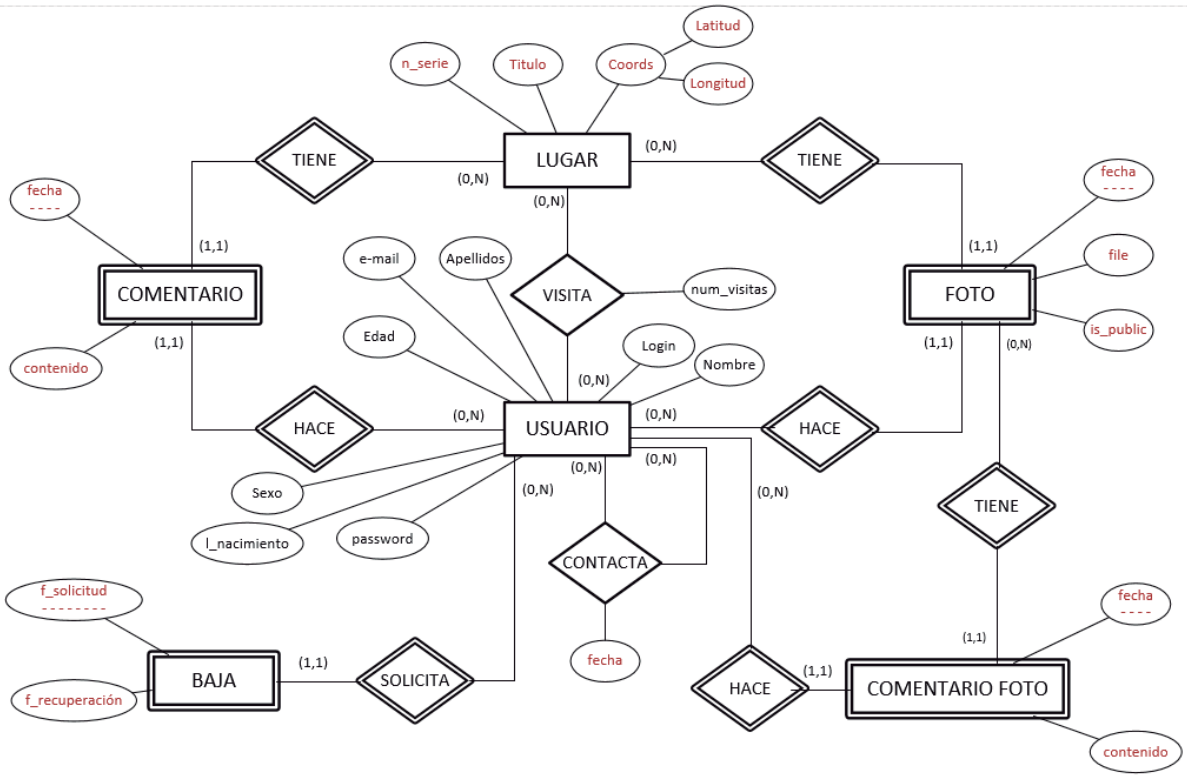


Fig. 7: Diagrama entidad relación de la base de datos

2. DISEÑO DE GESTIÓN DE APLICACIÓN

Para la aplicación en servidor usaremos una arquitectura en tres capas: capa de acceso a datos o persistencia, capa de gestión de contenidos y capa de servicios web. Lo hemos podido ver reflejado de manera más gráfica en la Fig. 6: Diagrama de arquitectura del sistema.

CAPA DE ACCESO A DATOS

Dadas las tecnologías de Microsoft® .NET Framework para el acceso a bases de datos conocidas como ADO.NET realizaremos un diseño acorde a las ventajas que nos proporciona esta API.

Para el acceso a datos usaremos lo que en ADO.NET se denomina modo desconectado. Consiste en copiar partes de la base de datos en memoria de manera que solo exista una conexión a la base de datos, y todas las peticiones y consultas de los usuarios se realicen sobre la copia en memoria. Este sistema aumenta considerablemente el rendimiento ante el acceso múltiple de usuarios al servidor, ya que es más rápido el acceso a la memoria que una consulta a la BD.

El sistema gestor por tanto solo admitirá conexiones de la aplicación de servicios, y será esta la que gestionará las múltiples conexiones de usuarios.

La implementación de los métodos de acceso se realizará como métodos estáticos de dos clases distintas según sea para control de usuarios o de contenido. A su vez no se podrán crear instancias de estas clases ya que no tiene sentido (solo métodos de clase no de instancia). Cada una de ellas tendrá una copia estática de la referencia del DataSet (copia en memoria de la base de datos.

El DataSet y los DataAdapters presentan toda la interfaz de acceso a la base de datos, el primero de ellos almacena la copia en memoria de los datos, mientras que el segundo proporciona la conexión efectiva a la base de datos.

Se ha optado por el uso de un patrón singleton para acceder al DataSet. Consiste en definir una clase factoría para que únicamente exista una instancia del DataSet controlando también, en la inicialización, la conexión a la BD. El diseño detallado de los ensamblados correspondientes (DataModule y DataModel) se pueden ver en: **Anexo I: Diseño – APS.LogrAR.DataModule**

CAPA DE LÓGICA DEL SISTEMA

El planteamiento de la capa de lógica se hace partiendo de que será manejada en el servidor y se accederá ella desde el cliente mediante servicios web. Por ello se diseña una interfaz orientada a objetos de la capa de lógica, separada según corresponda a acciones del sistema o a acciones que realiza el propio usuario. Quedarán definidas como dos interfaces ILogrARSystem e ILogrARUser.

Estas interfaces se implementarán tanto en el servidor como en el cliente.

En el servidor, la implementación de estas clases controlará la capa de acceso a datos para grabar y obtener datos del sistema. Además se encarga de proporcionar instancias de lógica de usuario, ya que se ha diseñado de tal manera que estas últimas por seguridad solo se pueden obtener con el método “Login” de la lógica de sistema. Posteriormente cada instancia de lógica de usuario estará relacionada con una sesión de servicios web.

En el cliente en cambio la implementación de lógica de negocio contactará con el cliente web que proporciona acceso a los servicios del servidor, y será el encargado de pedir los datos al servidor o al contrario mandarlos. También tendrá la función descrita en el servidor de controlar que solo se acceda a la lógica de usuario mediante un login valido. Esto abstrae el uso de las conexiones web haciendo que de cara a la capa gráfica presente la interfaz de lógica exactamente igual a la que tiene el servidor.

La clase de lógica de sistema realizará una función de factoría de lógica de usuario ya que como hemos mencionado es la encargada de proporcionarnos instancias de esta, controlando además para obtener una instancia que disponemos de un login correcto.

El uso de una interfaz de lógica unificada nos proporciona una API de servicios para manejo del sistema sobre la cual podremos construir GUIs, servicios web, etc. en distintas plataformas (móvil, PC, etc.). Ver diseño detallado en: **Anexo I: Diseño – APS.LogrAR.Management**

CAPA DE SERVICIOS WEB

La implementación de los servicios web consistirá en adaptar los servicios proporcionados por la capa de lógica de manera que sean compatibles con la transmisión de datos por medio de SOAP.

Al igual que la capa de lógica se separa en dos bloques, es decir dos servicios web diferentes, uno para uso de funciones propias de usuario (solo funciona con sesión iniciada), y un segundo para funciones globales del sistema. Este será el que nos proporcione la funcionalidad de acceder al sistema.

Respecto a la serialización de datos no primitivos no es necesaria ninguna actuación al respecto, ya que al únicamente serializar objeto del modelo de datos y listas funciona correctamente la serialización por defecto. Esta es una de las motivaciones de haber dejado las propiedades de estas clases públicas, ya que de lo contrario habríamos tenido que implementar la interfaz ISerializable en cada una de las clases.

De hecho, esto último habría sido lo óptimo para la implementación del sistema pero debido al marco en el que nos encontramos es inviable implementar esta interfaz en todos los objetos del modelo de datos. Principalmente por el tiempo que nos llevaría hacerlo.

Los servicios finales quedarán disponibles en un servidor IIS, donde podremos acceder a su descripción mediante WSDL y comunicarnos mediante SOAP, con lo que son accesibles desde cualquier plataforma.

También se incluye un tercer servicio que permite la subida de imágenes al servidor, este servicio está asociado al servicio de funciones de usuario, ya que es necesario tener una sesión activa y se complementa con este para subir fotos al sistema.

3. DISEÑO DE LA APLICACIÓN MÓVIL

El cliente móvil, al igual que el servidor lo vamos a estructurar en una arquitectura de tres capas, que se corresponderán con: capa de acceso a servicios (cliente web), capa de lógica (manejo de

los servicios y abstracción de estos a la capa gráfica), y capa gráfica (interfaz de usuario). Esto también queda reflejado en la Fig. 6: Diagrama de arquitectura del sistema.

CLIENTE DE SERVICIOS WEB

La aplicación móvil se comunicará con el servidor mediante un módulo cliente de servicios web. Este módulo será el encargado de hacer las peticiones e interpretar las respuestas del servidor, se implementará mediante tres clases generadas por la utilidad WSDL de Microsoft .NET Framework. Esta utilidad nos crea una clase con los métodos necesarios para acceder al servicio web ahorrándonos el trabajo de acceder a los servicios de forma manual.

Para el correcto funcionamiento una vez generadas será necesario adaptarlas para el uso de nuestro modelo de datos y hacerlas compatibles con el uso de cookies ya que mantienen sesiones abiertas.

Por tanto una vez implementadas tendremos acceso a los métodos de los servicios web definidos.

MANEJO DE LÓGICA EN CLIENTE

El segundo paso para terminar la comunicación del cliente con el servidor es adaptar estos métodos para que correspondan con la interfaz de lógica definida. Implementaremos las interfaces de lógica de manera que llamen por debajo a los servicios web, quedando la misma capa de lógica en cliente y servidor.

Una vez hecho esto, definimos como se accederá a la lógica desde la aplicación. Para ello usaremos una factoria que contendrá una instancia de la lógica de sistema, y, si la sesión está abierta, una instancia de lógica de usuario. De no haber iniciado sesión no dispondremos de una instancia de lógica de usuario por lo que nos será imposible acceder a sus métodos ya que no es instanciable. Sólo podemos crear este acceso registrándonos en el sistema.

DESARROLLO API GRÁFICA SOBRE EL ENTORNO UNITY

Para la construcción posterior de la interfaz de usuario se va a desarrollar un API de GUIs sobre Unity, la cual se orienta de la misma forma que podemos ver la mayoría de APIs de este tipo. Es decir partimos de un controlador para el manejo de vistas, en las cuales tenemos diferentes controles con un comportamiento u otro según la función que tenga esta vista. Las acciones del usuario se manejan mediante eventos que son lanzados según las entradas del usuario. Se incluye en la Fig. 8 el diseño completo de la API mediante un diagrama de clases.

Se ha estudiado la integración de este API en Unity de manera que sea lo más transparente posible, por lo que se ha llegado a la siguiente solución.

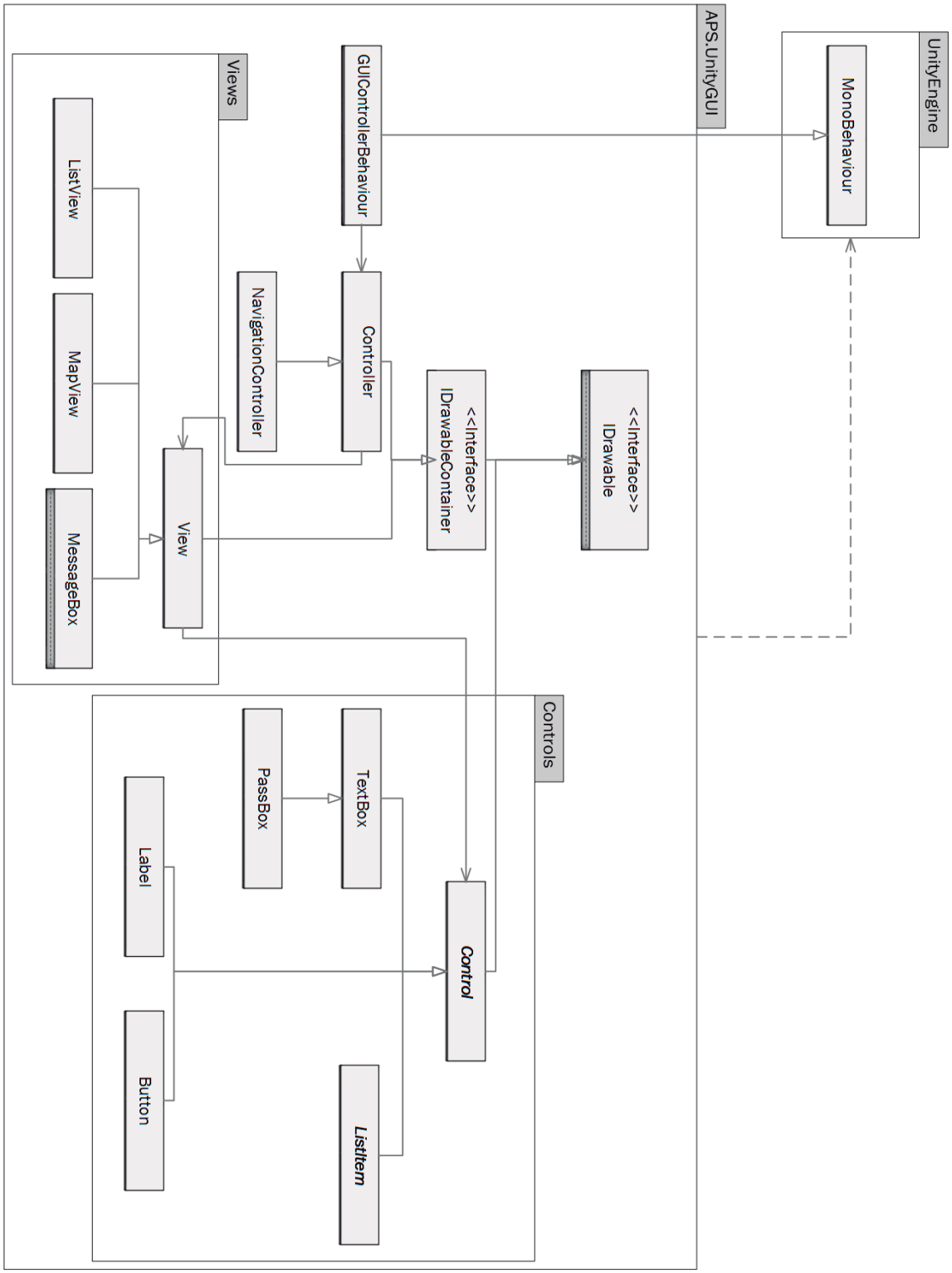


Fig. 8: Diagrama de clases de la API desarrollada: Ensamblado APS.UnityGUI

En Unity no tenemos un flujo de programa habitual, si no que cada uno de los componentes de una escena pueden tener asociados scripts. Cada uno de ellos posee dos métodos para manejar ese objeto según lo programemos. Pero hay un objeto especial que es el objeto cámara, éste además de controlar el objeto tiene la propiedad de poder dibujar GUIs. En el fondo cada uno de estos scripts son clases que heredan del manejador principal (MonoBehaviour) que es el que permite que se ejecuten.

Por ello nuestra API será gestionada mediante la clase GUIControllerBehaviour, ésta clase hereda de la clase UnityEngine.MonoBehaviour. De esta manera si nosotros en uno de estos scripts heredamos de GUIControllerBahaviour en vez de MonoBehaviour automáticamente se nos creará un controlador sobre el cual podemos instanciar vistas y controles que se dibujarán en la GUI. Pero dado que el único componente gráfico de Unity que admite capa GUI son los de tipo cámara solo podremos usar clases heredadas de GUIControllerBehaviour en objetos de tipo cámara.

INTERFÁZ DE USUARIO

Como cabía esperar, todo el manejo de la interfaz de usuario de la aplicación la realizará el API que hemos diseñado para la administración de GUIs en Unity, por lo que las vistas que nosotros usaremos estarán basadas en este API.

En primer lugar definimos la funcionalidad de la interfaz en relación con lo especificado para el paquete de trabajo 1.02.03 Implementación interfaz usuario. Por lo que se satisfarán los casos de uso: registro de usuario, añadir fotos, añadir visitas, añadir comentario, ver comentarios, ver fotos y añadir contactos. Además también presentará las indicaciones necesarias para encontrar cada uno de los puntos de interés y cuando estemos en ellos un acceso al contenido asociado en cada uno.

Quedando claras las funciones que va a incluir la interfaz se diseña la navegabilidad de esta de forma que sea limpia y coherente para su uso por un usuario medio. Podemos ver un diagrama de navegabilidad en la Fig. 9 donde se muestran los nombres de cada una de las vistas enlazadas según como accedemos de unas a otras.

Para el diseño gráfico tenemos la ventaja de que al usar nuestra API con Unity podemos personalizar todo lo que queramos nuestra interfaz, ya que podemos cambiar el look and feel de cada componente sin problemas.

Teniendo en cuenta esto vamos a realizar un diseño inspirado en el funcionamiento del sistema operativo iOS pero con personalizaciones a nuestro gusto. Como comienzo se esbozan a mano

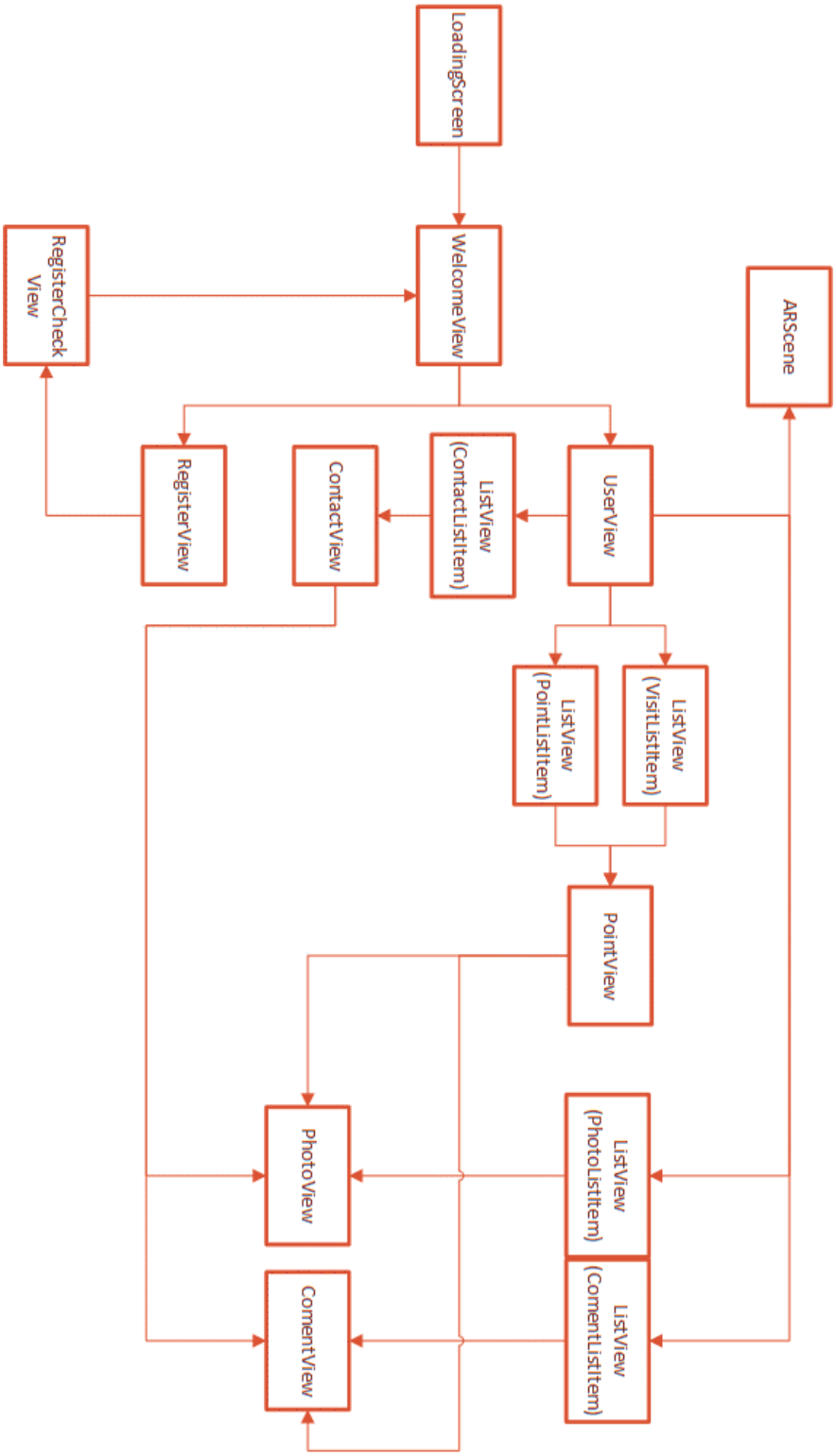


Fig. 9: Diagrama de navegabilidad entre las vistas de la aplicación. Cada una de estas se identifica con el cuadro con su nombre correspondiente.

alzada los diseños de cada una de las vistas incluidas en Fig. 9, podemos encontrar estos bocetos en **Anexo I: Diseño – Vistas de aplicación**.

Una vez realizados los bocetos, identificamos qué texturas necesitamos para diseñarlas a ordenador y generar el paquete de texturas correspondiente con el que luego trabajaremos. Ejemplificamos gráficamente este proceso en la Fig. 10. En nuestro caso hemos generado 33 texturas para el diseño completo de la interfaz que podemos ver en la Fig. 11. Todas ellas han sido realizadas para este proyecto, y quedan ligadas a la licencia a la que se adhiere el proyecto.

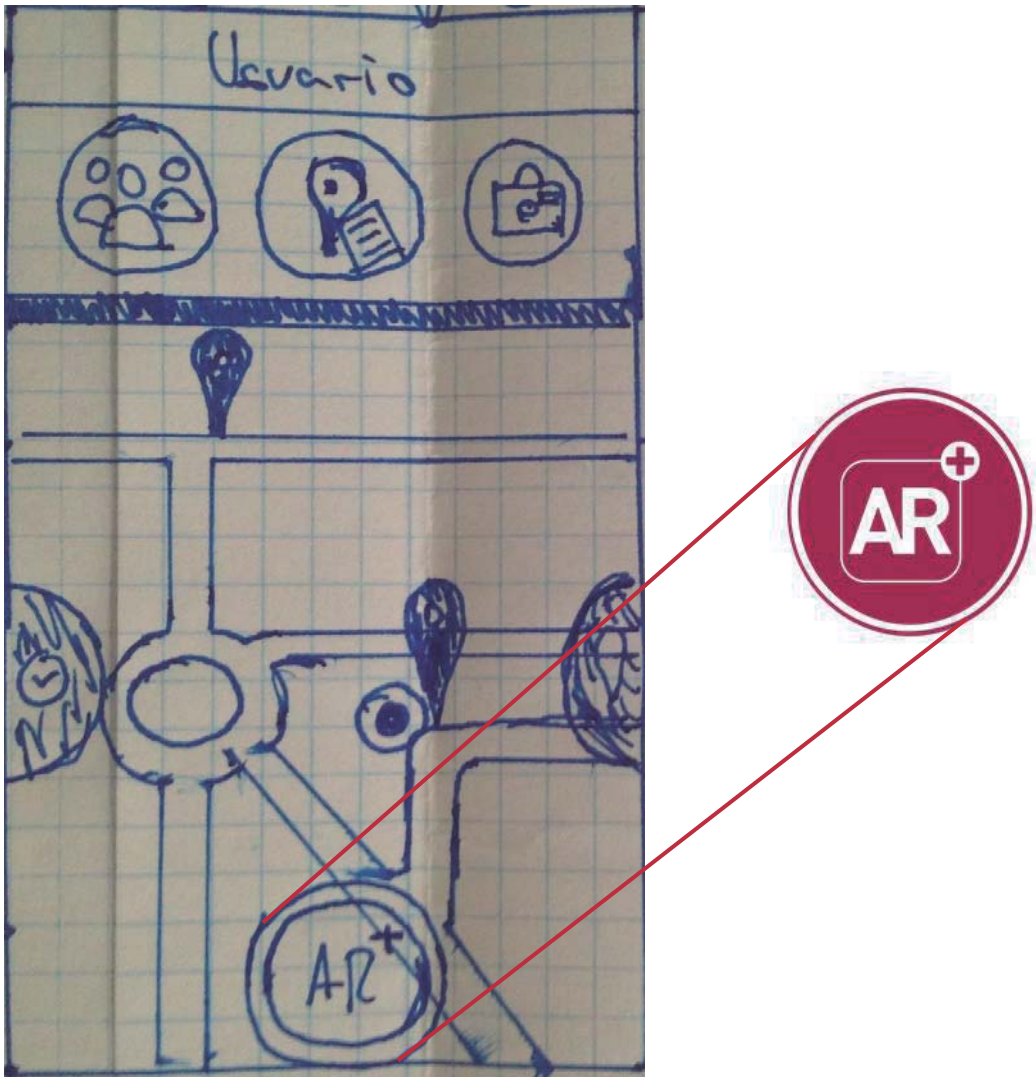


Fig. 10: Textura desarrollada a partir de uno de los bocetos de la interfaz

Respecto al diseño del software cada vista heredará de la clase APS.UnityGUI.View y redefinirá los métodos que precise, así como los elementos ListItem heredarán de esta clase haciendo lo

propio. Todo el comportamiento de navegabilidad estará delegado en el "NavigationController" de la API.

Fig. 11: Paquete de texturas de la aplicación

La interfaz para el acceso a la lógica de la aplicación usará la factoría LograrAccesFactory mediante la cual obtiene las instancias de lógica de sistema y de usuario que corresponden en cada momento.

IMPLEMENTACIÓN

1. IMPLEMENTACIÓN DE GESTIÓN DE APLICACIÓN

CAPA DE ACCESO A DATOS

Tras implementar la base de datos en SQL Server mediante los Scripts añadidos en el **Anexo 2: Implementación** se genera un DataSet personalizado para la base de datos que denominamos LograrDataSet.

Esto permite tener un fácil acceso y control de la base de datos, un ejemplo de su funcionamiento:

```
LograrDataSet lds=new LograrDataSet();
Lds.USERS.FindByUsrname("unirioja")
```

Como vemos nos abstrae de consultas y nos proporciona métodos para realizar búsquedas automáticas. Además de modifica el código generado para:

- Proteger el constructor (patrón singleton, uso de fatoría)
- Modificar el método AcceptChanges para que lance un evento definido "ChangesAcepted" de tipo "CommitEventHandler" que es un delegado definido por nosotros para controlar la acción de commit.

Tras ello en la factoría que usaremos para obtener la instancia del DataSet asignamos a ese evento un método respuesta cuya finalidad es actualizar la base de datos si los datos han sido modificados. Sólo actualiza las filas modificadas desde que se cargó o desde la última actualización.

Además la factoría se encarga de cargar el DataSet cuando es instanciado. De manera que el DataSet queda protegido para que sólo exista una instancia de él (constructor definido como de ensamblado, no podeos acceder desde fuera). Y controla que al inicializar el DataSet siempre cargue los datos antes de devolver la referencia.

Otro aspecto a destacar es que algunas de las búsquedas necesarias eran poco eficientes al ejecutarlas sobre el DataSet. Para solventar esto se han añadido como procedimientos almacenados que se llaman desde la aplicación, podemos verlo en detalle en el script de "storedProcedures" en el **Anexo 2: Implementación – Base de datos**

La implementación del control de contraseñas, se ha realizado mediante el uso de hash MD5 y se ha llevado a cabo de forma que la aplicación nunca los obtiene de la base de datos, solo

compara el que ha calculado la contraseña suministrada con el existente y devuelve si coinciden. De esta forma se protege sobre ataques contra el hash de la contraseña.

El control de contenido y enlaces a recursos se hace combinando la información obtenida en la BD con la información contenida en el archivo de configuración de la aplicación. En este se configuran los paths donde están alojados los recursos en el servidor.

LÓGICA DEL SISTEMA (SERVIDOR)

La implementación de lógica de servidor se hace mediante las clases LogrARSystem y LogrARUser. Ambas implementan la interfaz de lógica correspondiente. Además la segunda de ellas hereda de APS.LogrAR.DataModel.User. Es decir a partir del usuario definido en el modelo de datos le proporciona funcionalidad y lógica manteniendo los datos propios del usuario correspondiente.

Por tanto existirán tantos objetos de tipo LogrARUser como usuarios conectados al sistema, ya que estos objetos estarán también ligados a la sesión del usuario, y representan a un usuario, proporcionándole las acciones que tiene disponibles dentro del sistema.

Para el manejo de login se realizan hash de los password mediante un algoritmo MD5 el cual es gestionado por la clase MathMod en el paquete Utils. Se encarga de ello esta capa ya que por seguridad la capa de persistencia solo trabaja con los hash, no con los password.

Se implementa también una conexión a servidor de correo para el envío de códigos de recuperación de contraseña. Modulo Mail del paquete Utils. El código lo almacena el sistema en memoria durante un tiempo, luego se borra por lo que no es almacenado en la base de datos en ningún momento.

La gestión de errores se realiza mediante una excepción definida “LograrError”. Esta excepción ya únicamente alacena el mensaje de error del usuario, no contiene información de depuración. Toda ella es encapsulada para que solo se puedan propagar este tipo de errores a la siguiente capa protegiendo así la información sobre cómo está construido el sistema.

SERVICIOS WEB

La implementación de los servicios en .NET se realiza en el ensamblado “APS.LogrAR.WebServices”. Al igual que en otras plataformas cada servicio es una clase que hereda de “WebService” e implementa métodos web, marcándolos como “WebMethod”.

En nuestro caso hemos tenido que añadir a la implementación la función de habilitar la sesión. Con el fin de cuando un usuario se registra mantener su sesión abierta en el sistema. De esta manera se realiza la gestión del login. Cuando un usuario se registra crea una sesión y se guardan

sus datos en el scope "Session". En posteriores conexiones a esta sesión se buscan los datos en la sesión, de no estar se interpretará que el usuario no está registrado.

De hecho, de los tres servicios que se implementan, el correspondiente al usuario requiere tener un usuario con sesión abierta para poder trabajar, de lo contrario nos dará un error de sesión no iniciada.

2. IMPLEMENTACIÓN DE LA APLICACIÓN MÓVIL

CLIENTE DE SERVICIOS WEB

Tras generar las clases cliente de los servicios web es necesario adaptarlas para el uso de nuestro modelo de datos, ya que la utilidad en vez de tomar nuestro ensamblado crea unas estructuras de datos con el mismo contenido pero distintas. El problema se resuelve eliminando estas estructuras definidas por la utilidad y agregando una referencia al namespace del modelo de datos, con lo que ahora nos devuelve objetos de nuestro modelo de datos compatibles con el resto de la lógica de aplicación.

También para la gestión correcta de las sesiones tenemos que añadir el manejo de cookies. Para ello modificamos las clases para que integren por defecto un "CookieContainer". Además debemos de gestionar que todos los servicios usen el mismo "CookieContainer" ya que de lo contrario no se reconocería la sesión de un servicio a otro.

LÓGICA DE APLICACIÓN

La implementación de la interfaz lógica se hace adaptando los métodos de los clientes a sus correspondientes métodos de lógica.

Se gestionan también los errores ya que estos son generados en el servidor y encapsulados en una excepción de tipo SoapException que debemos tratar para acceder al error que encapsulan. Una vez obtenido el error es lanzado como un LograrError para que la capa gráfica lo gestione más adelante.

La factoría LograrAccesFactory se impleenta con dos propiedades estáticas que nos devuelven un objeto LogrARUser o LogrARSystem según corresponda, y un método Login. Este método llama al método login de lógica del sistema y registra al usuario en la factoría, hasta ese momento la propiedad *USER* devolverá null, por lo que no tendríamos acceso a los métodos de usuario.

DESARROLLO API GRÁFICA SOBRE EL ENTORNO UNITY

La implementación del API se hace siguiendo los diseños detallados anteriormente. Además se incluyen en el paquete APS.UnityGUI los delegados correspondientes a los eventos del usuario.

Todos los eventos son capturados por el controlador (reconoce el tipo de evento que debe lanzar en todo momento) y lanzados al objeto que le corresponda el evento, con lo cual también es el encargado de delegarlo a la vista correspondiente y esta de lanzarlo al componente correspondiente en el que podremos haber asignado un manejador al evento.

La vista MessageBox es un tipo de vista especial ya que su función es mostrarnos cajas de texto informativas, no podemos añadir controles en ella ni instanciarla, únicamente consta de métodos estáticos para mostrar y ocultar ventanas de mensaje.

Desde la clase GUIControllerBehaviour podemos acceder al controlador raíz de la aplicación mediante una propiedad estática para añadir, modificar o navegar vistas.

Unity solo nos da información sobre los toques que se producen en la pantalla por lo que en el controlador se implementan algoritmos que a partir de la detección de toques en pantalla, contado de estos, su posición y demás información, detectan que evento se debe lanzar y hacia qué componente lo debe de propagar. Estos algoritmos han sido diseñados específicamente para ello, podemos encontrar su especificación en **Anexo 2: Implementación – Algorítmos**

Para especializar cada uno de los controles se sobrecarga su método FrameDraw. Este es el que dibuja la interfaz y hereda de “IDrawable”. Además, para los controles, implementamos la clase abstracta “Control” que engloba muchas propiedades comunes de los distintos controles.

Las distintas vistas heredan de la clase View que se encarga de gestionar el comportamiento de la interfaz, y de manejar entradas del usuario como el deslizar el dedo por la pantalla para avanzar por los contenidos, etc. A esta será a la que añadiremos controles. Podemos ver los tipos de vista implementados en Fig. 8: Diagrama de clases de la API desarrollada: Ensamblado APS.UnityGUI.

Además cabe mencionar la implementación de la clase “NavigationController” la cual hereda la funcionalidad de “Controller” y añade navegabilidad entre vistas, haciendo transparente su gestión. Tras registrar la primera vista, crea una pila vacía, donde guarda la vista cargada si le indicamos que navegue a una nueva. Cuando retrocedemos va desapilándolas y volviéndolas a cargar en pantalla.

Todo ello es transparente al usar la API, al asociar vistas al controlador navega a ellas y al hacer “Back()” regresa a la anterior.

INTERFÁZ GRÁFICA DE USUARIO

Para el desarrollo de la interfaz gráfica de usuario hemos usado el API gráfica desarrollada anteriormente.

En primer lugar decir que la interfaz de usuario fuera de los contenidos de realidad aumentada se realiza en una escena Unity[1] separada y vacía por los siguientes motivos:

- Reducción de elementos en memoria.
- No es necesario estar procesando la entrada de cámara
- Permite implementar cada lugar de las aplicación en escenas distintas para mejorar el rendimiento

Esta escena constara solo de un objeto cámara simple con su GUI activada. A esta cámara asociaremos un manejador heredado de nuestro "GUIControllerBahaviour" para activar el uso de nuestro API. Este manejador configurará las URL de acceso a los servicios web iniciando la API cliente web, y creara una instancia de "WelcomeView". El API reconocerá esta instancia y entrará en funcionamiento delegando ya el comportamiento en la vista. A partir de este momento cada vista es la que gestiona el contenido y funcionalidad que le corresponda accediendo a la capa de lógica cuando sea necesario. También cada una de ellas decide a qué vista debe navegar según la entrada del usuario.

A fin de mejorar el rendimiento y la velocidad de respuesta de la aplicación todo el contenido dinámico se carga en paralelo evitando retardos en la respuesta de la aplicación. Esto se hace mediante el lanzamiento de hilos para tareas que pueden tener un retardo a causa de la red (tareas que requieran de conexión a internet).

Para paliar la memoria que ocupamos con todo este contenido desarrollamos un sistema de recolección de contenido que no vamos a usar. Cada vez que volvemos atrás a la vista principal, o abrimos la característica "AR" se limpian los recursos cargados y que no estén usándose.

Con la combinación de estas dos características (threading y control de cache) generamos una aplicación estable y a la vez rápida en la respuesta al usuario.

Por último añadimos un sistema de cache de los datos de usuario, ya que dado que es una aplicación móvil, normalmente siempre la usara la misma persona. Por tanto este sistema se encargará de al cerrar y volver a abrir la aplicación recordar sus datos de login y abrir una sesión con ellos.

Se implementa usando SQLite y un pequeño gestor dentro del ensamblado APS.LogrAR.Management.Client que nos facilita el acceso al fichero de datos.

[1] Unity gestiona el contenido mediante escenas independientes entre sí. Cada escena contiene objetos, comportamientos, etc. En el momento que se carga una de ellas se descarga la anterior si la hay.

IMPLEMENTACIÓN DEL CONTENIDO EN REALIDAD AUMENTADA

Cuando cargamos el contenido en AR la aplicación detecta el punto en el que estamos y carga la escena correspondiente a ese punto. Cada escena se implementa específicamente para cada punto ya que todos ellos tienen contenido diferente.

Estas escenas muestran algunos elementos de interfaz gráfica (comunes entre ellas) para su manejo que son gestionados de manera similar a la interfaz global con nuestro API. Pero además implementan el comportamiento de los objetos que cargan sobre el mundo real. Todo este comportamiento se programa en scripts dentro de cada uno de los objetos, luego pueden ser manejados desde los objetos de la GUI para cambiar su comportamiento según las entradas del usuario.

Como especificamos en el paquete de trabajo 1.02.04 solo implementamos completamente 3 de estas escenas a fin de poder mostrar el potencial de la aplicación sin superar el número de horas disponible.

El contenido añadido ha sido obtenido en la red, mediante un proceso de búsqueda de información. Todo él está sujeto a licencias Creative Commons de libre uso, a excepción del avatar usado en la escena “Estatua de Espartero” que ha sido desarrollado junto con la empresa “CreativiTIC” y por tanto sujeto a los términos de este proyecto. Todo el está contenido en la aplicación a excepción de los videos, los cuales son reproducidos a través de internet desde el portal youtube. Para la carga de estos se usa una librería desarrollada durante el periodo de prácticas por lo que no es detallada en el proyecto.

Los marcadores correspondientes a cada una de las escenas (objetos donde se reconoce contenido) están en el **Anexo III: Marcadores**.

DESPLIEGUE Y PRUEBAS DEL SISTEMA

1. APLICACIÓN DE SERVICIOS WEB (PARTE SERVIDOR)

DESPLIEGUE

Para desplegar los servicios necesarios para el funcionamiento de la aplicación usamos en un comienzo un servidor virtual (máquina virtual dentro de nuestro equipo de trabajo). En él de acuerdo con el desarrollo que hemos realizado desplegamos un entorno de Windows Server 2008, configurando los servicios que vayamos a necesitar. En nuestro caso se instala y configura una instancia de SQL Server 2012 y de IIS 7, quedando con ello dispuestas todas las herramientas necesarias para el despliegue de nuestros servicios.

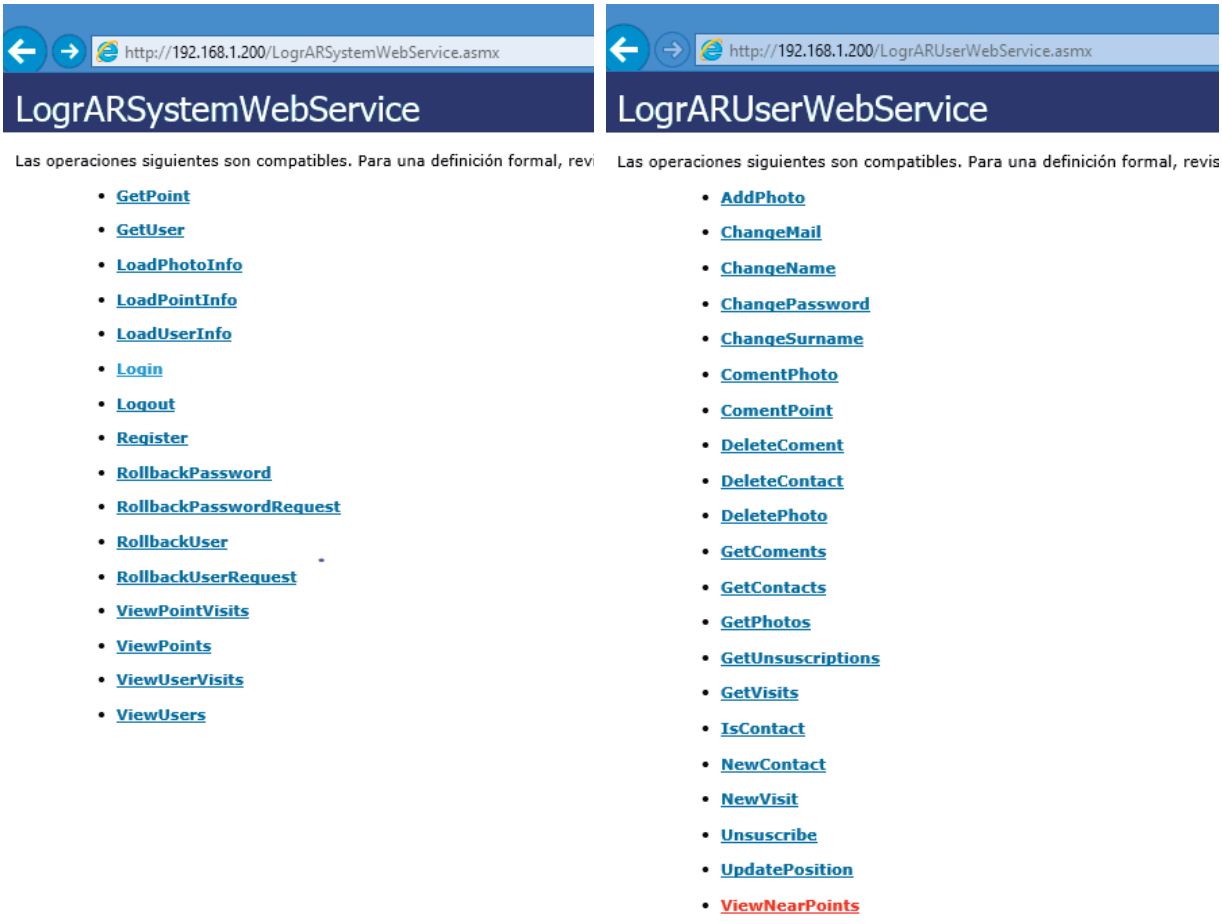


Fig. 12: Vista de los servicios desplegados en IE

Para comenzar desplegamos la base de datos con el script de creación que hemos realizado y podemos ver en el **Anexo II: Implementación - Base de datos**, además creamos un usuario con

los permisos restringidos a aquellos que sean necesarios para el manejo de la aplicación. Con este será con el que la aplicación de gestión realice las conexiones.

Teniendo ya en funcionamiento la base de datos procedemos al despliegue de los servicios web, para ello nos vamos a ayudar de la herramienta "WebDeploy" de Microsoft con la cual podemos desplegar la aplicación compilada en VisualStudio importando el archivo generado. Una vez desplegada modificamos el fichero "web.conf" añadiendo los datos de conexión a SQL Server y la dirección de la carpeta donde almacenaremos el contenido multimedia de la aplicación. También debemos configurar IIS para que sirva el contenido de esta carpeta como contenido web. Tras ello reiniciamos el servidor y probamos que los servicios se están ejecutando como vemos en la Fig. 12.

Tras realizar este despliegue surge la posibilidad de a través de la Universidad de La Rioja conseguir un servidor dedicado y siempre accesible para el uso a lo largo del proyecto. Por ello se determina sustituir el servidor usado hasta ese momento por el facilitado por la universidad.

Para acceder a este servidor debíamos de cumplimentar varios documentos en primer lugar para la solicitud del servidor, y en segundo lugar para solicitar acceso a la VPN de la universidad. Estos documentos tuvieron que ser firmados por el tutor para permitirse el acceso del alumno a la red interna de la universidad.

Una vez conseguido el acceso al servidor se realizó el mismo proceso descrito al comienzo de este apartado con el añadido de restaurar los datos que teníamos en el primer servidor. Lo cual se hace de manera sencilla con un backup de la base de datos. Como añadido en este servidor ha sido necesario habilitar SSL para poder acceder a los servicios debido a la estricta política de seguridad de la universidad. Esto ha supuesto el generar un certificado autofirmado para el servidor que es instalado en el terminal de pruebas permitiéndole establecer la conexión SSL. Para un despliegue definitivo de la aplicación deberíamos solicitar un certificado SSL válido a una autoridad de certificación.

PRUEBAS

Para la realización de las pruebas se ha usado el servidor final de la aplicación, es decir el suministrado por la universidad. Para realizarlas de manera cómoda hemos usado unos formularios de prueba que genera IIS para probar los servicios e introducir daos en cada método. Los aspectos que vamos a considerar en esta prueba son los siguientes.

- Obtención de información completa, sin errores y acorde a la solicitud
- Guardado correcto de datos y datos cifrados
- Gestión correcta de la sesión HTTP cuando es necesario

- Gestión adecuada de los errores cuándo esto se producen
- Robustez ante la entrada de datos aleatorias e incoherentes.

Tras finalizar las pruebas de los métodos web uno a uno, se dan por satisfechos los requisitos anteriores, quedando por validada la aplicación de servicios web.

2. APLICACIÓN MÓVIL (PARTE CLIENTE)

DISTRIBUCIÓN DE LA APLICACIÓN (DESPLIEGUE)

Terminada la implementación de la aplicación procedemos a generar las distribuciones que quedarán disponibles para los usuarios. Como ventaja del uso del entorno Unity3D tenemos la posibilidad de generar la aplicación para las plataformas Android e iOS.

Una vez generadas las aplicaciones, “apk” para Android (directamente instalable en dispositivos), y proyecto XCode para iOS (necesaria su compilación en la plataforma de Apple) procedemos a su instalación.

En nuestro caso como el desarrollo se ha orientado hacia los dispositivos de Apple, será esta distribución la que tomaremos como referente en las pruebas. Pudiendo la distribución Android no satisfacer las condiciones probadas ya que como se definió en el alcance no sería optimizada en este proyecto.

PRUEBAS DE LA APLICACIÓN

Realizamos la instalación de la aplicación en un iPhone 5 compilando el proyecto de XCode para después lanzar la aplicación. En la Fig. 13 podemos ver el aspecto que presenta finalmente la aplicación, como vemos acorde a los bocetos del diseño mencionados en el apartado de diseño. Tras probar todos los casos de uso implementados por la interfaz y el funcionamiento del contenido en realidad aumentada, damos por válida la aplicación ya que cumple los requisitos que habíamos definido sin fallos ni errores.

A pesar de no haber sido optimizada para la plataforma Android vamos a realizar la instalación en un dispositivo para realizar las mismas pruebas y comprobar qué tareas podrían mejorar la integración a Android. Tras ello se detectan las siguientes carencias:

- Errores en el guardado de datos. Al usar distinta plataforma Android no encuentra bien el directorio en el que guardamos la información cacheada, como son los datos de usuario, etc.
- Errores en el dibujo de la interfaz de usuario tras girar la pantalla del dispositivo

La subsanación de estos errores podría ser propuesta como mejora del proyecto actual.

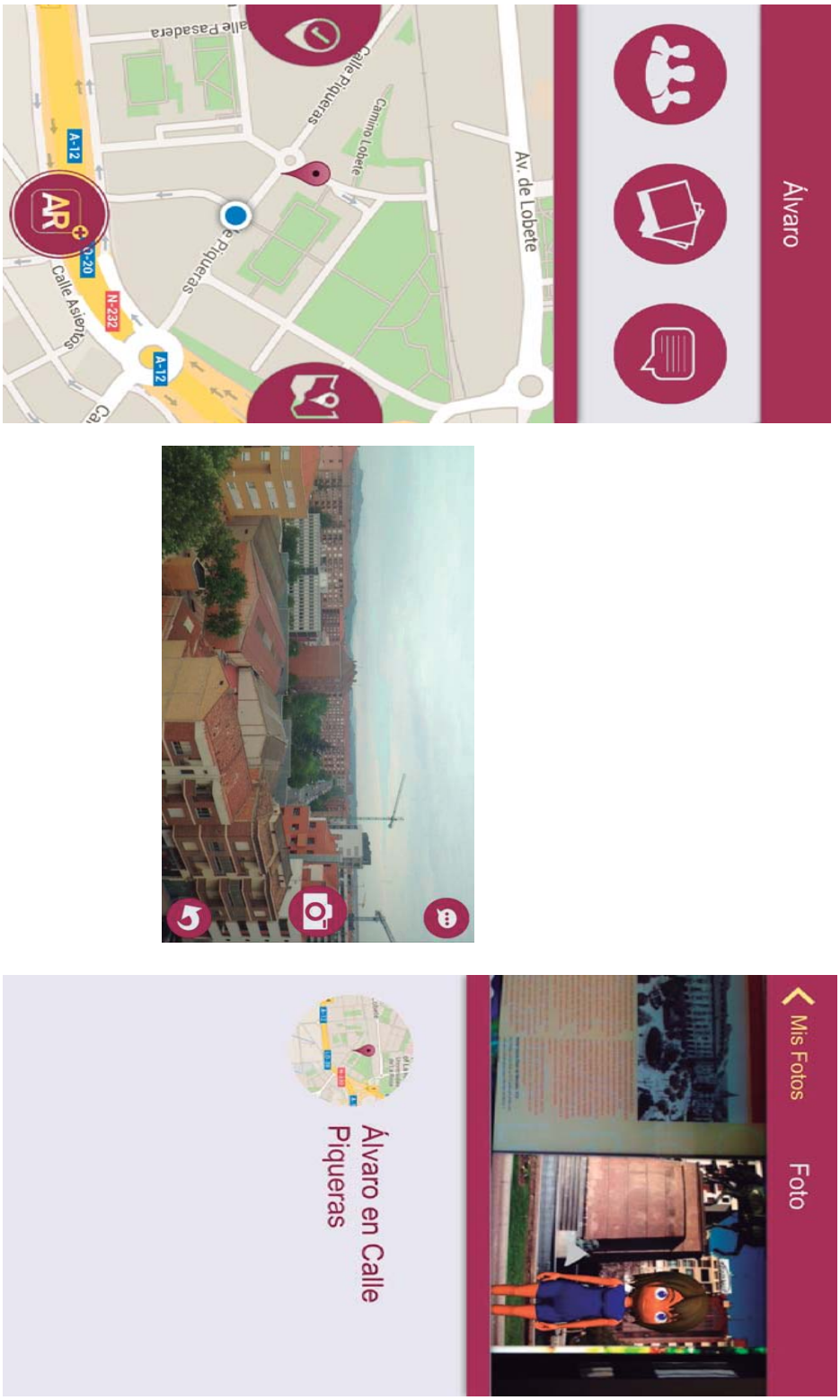


Fig. 13: Muestra de algunas pantallas de la aplicación final

SEGUIMIENTO Y CONTROL

1. SEGUIMIENTO DE TAREAS

- **Despliegue del entorno en el servidor** – Completado en tiempo previsto
- **Diseño e implementación de la base de datos** – Completado en tiempo previsto
- **Implementación parte de gestión de BD** – Completado en tiempo previsto
- **Implementación parte de gestión de contenido** – Completado en tiempo previsto
- **Implementación parte de lógica y servicios** – Completado con adelanto

El tiempo empleado en el estudio de los servicios web y su implementación fue menor del estimado por lo tanto la tarea fue concluida en un tiempo menor que el estimado. A pesar de haber podido adelantar considerablemente el trabajo con 10 horas de ventaja 5 de ellas se perdieron en resolver un problema con la serialización de objetos en los servicios web por lo que nos queda un balance positivo de 5 horas respecto a la planificación inicial

- **Pruebas de funcionamiento del servidor** – Completado en tiempo previsto
- **Solicitud y migración del servidor web –** Competado en más tiempo del previstio

Se emplearon dos horas más debido a la gestión necesaria para obtener el acceso a la red interna de la universidad. Fueron necesarios varios correos

- **Diseño de interfáz de usuario y recopilación del material necesario** – Completado en el tiempo previsto
- **Recogida de material e información** – Realizada en dos horas menos de lo previsto

Se planificaron ocho horas para esta tarea de las cuales solo hemos empleado seis. Esto se debe a que la recogida de material fue más rápida de lo esperado.

- **Implementación de la interfaz de usuario** – Completado en el tiempo previsto
- **Implementación de contenidos en realidad aumentada** – Completado en el tiempo previsto
- **Integración del sistema** – Completado en el tiempo previsto
- **Pruebas del sistema completo** – Realizadas en menor tiempo del previsto

Para la realización de las pruebas se esperaba que el sistema diese más errores de los que dio en estas y que habría sido necesario subsanarlos. Afortunadamente no fue necesario resolver problemas graves con lo que las pruebas solo ocuparon un marco de 8 horas.

- **Memoria, seguimiento y control del proyecto** – Se han empleado más horas de las previstas

Al ser necesaria una re planificación, junto con varios cambios importantes en el desarrollo de la memoria del proyecto se ha empleado más tiempo del previsto en el seguimiento, control del proyecto y desarrollo de la memoria. El tiempo empleado para ello ha sido finalmente de 25 horas distribuidas a lo largo del desarrollo del proyecto.

- **Planificación del proyecto** – Se emplea más tiempo del estimado.

Con motivo de la re planificación efectuada a lo largo del proyecto se incrementan en dos las horas usadas para la planificación del proyecto.

- **Desarrollo de material para la defensa** – A completar en la semana posterior a la entrega, completada la grabación del video demostrativo para incluirlo en la memoria.

Dados los retrasos en algunas de las anteriores tareas quedan tan solo 8 horas disponibles de las 300 iniciales para realizar esto, de las cuales hemos invertido 4 en la grabación del video. Por tanto quedan 4 horas disponibles para preparar la presentación y defensa. De lo contrario superaremos las asignadas a esa tarea.

Tareas	Tiempo estimado	Tiempo dedicado
Despliegue del entorno en el servidor	4	4
Diseño e implementación de la base de datos	6	6
Diseño e implementación gestión BD	12	12
Parte de gestión de contenido	28	28
Parte de gestión de lógica y servicios	65	60
Pruebas de funcionamiento de servidor	5	5
Solicitud y migración servidor web	0	6
Diseño de interfaz y confección de material necesario	16	16
Recogida de material e información	12	10
Implementación de la interfaz de usuario	48	48
Implementación de realidad aumentada	40	40
Integración del sistema	14	14
Pruebas del sistema completo	15	13
Seguimiento y control del proyecto	20	25
Planificación del proyecto	5	7
Desarrollo de material para la defensa	10	4 (Sin concluir)
TOTAL	300	300

Fig. 14: Tiempos dedicados/estimados de cada tarea

Como podemos ver en la Fig. 14 hemos empleado ya todo el tiempo disponible para la elaboración del proyecto sin haber terminado la tarea "Desarrollo del material para la defensa" que se estima que se empleen otras 4 horas de tiempo. Por tanto podemos afirmar que se concluye el proyecto con 304 horas de trabajo superando únicamente en 4 las horas definidas en un principio.

2. CONTROL

Durante la semana 6 del proyecto surge la posibilidad de solicitar un servidor dedicado a la universidad para tener disponibles de manera continua los servicios web de la aplicación.

Por tanto es necesario realizar una re planificación para dar cabida a las tareas necesarias para la solicitud y migración del servidor. Podemos ver el nuevo cronograma en la Fig. 15, Fig. 16, y un desglose de las tareas modificadas a continuación.

TAREAS MODIFICADAS

- **Diseño de interfaz y confección de material necesario –** Se ha reducido el tiempo de esta tarea en 6 horas, y replanteado para que se realice de forma paralela a la implementación con el fin de alternar diseño e implementación en la fase de desarrollo de la interfaz.
- **Recogida de material e información –** Se ha reducido el tiempo en 4 horas.

TAREAS AÑADIDAS

- **Solicitud y migración servidor web –** Se asignan 10 horas al proceso de solicitud y migración del servidor web

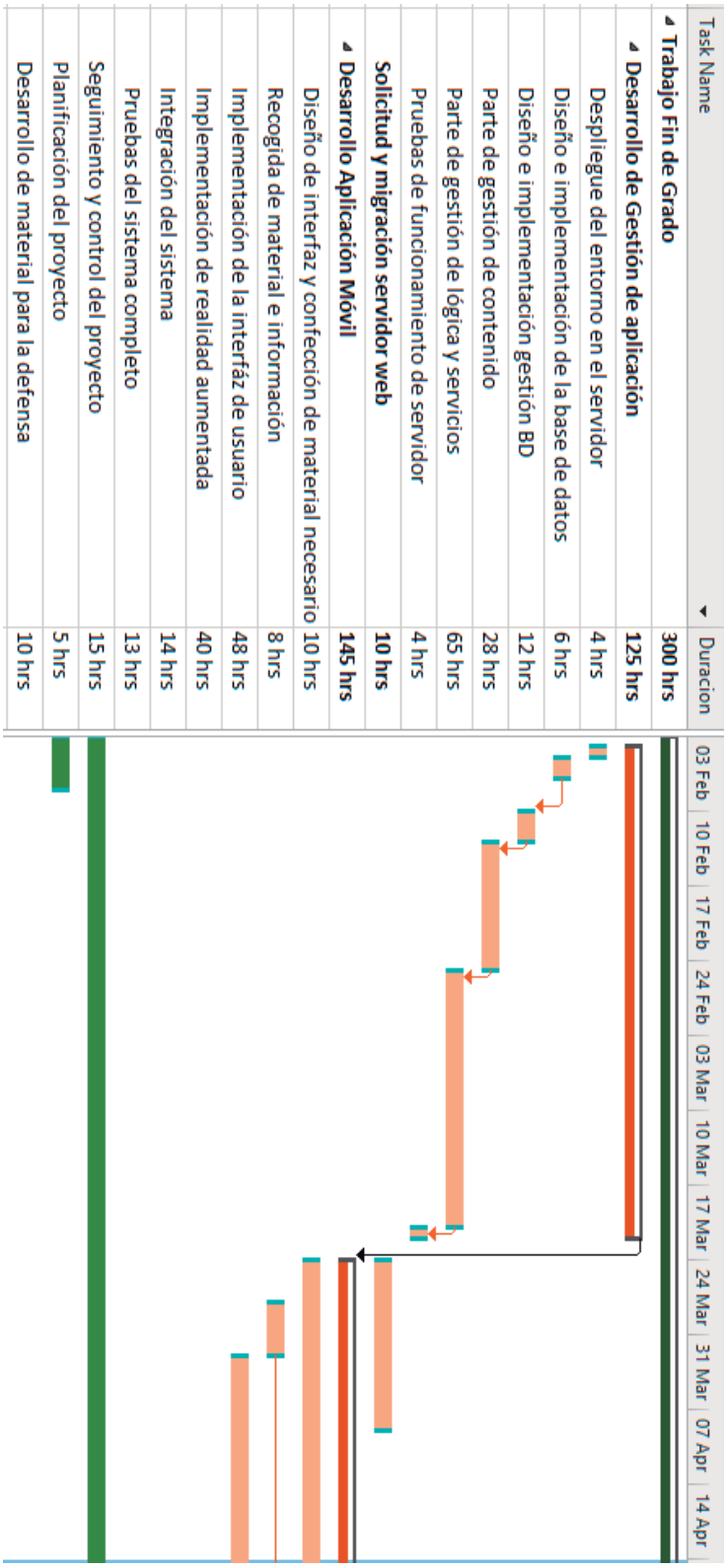


Fig. 15: Diagrama de Gannt 1/2

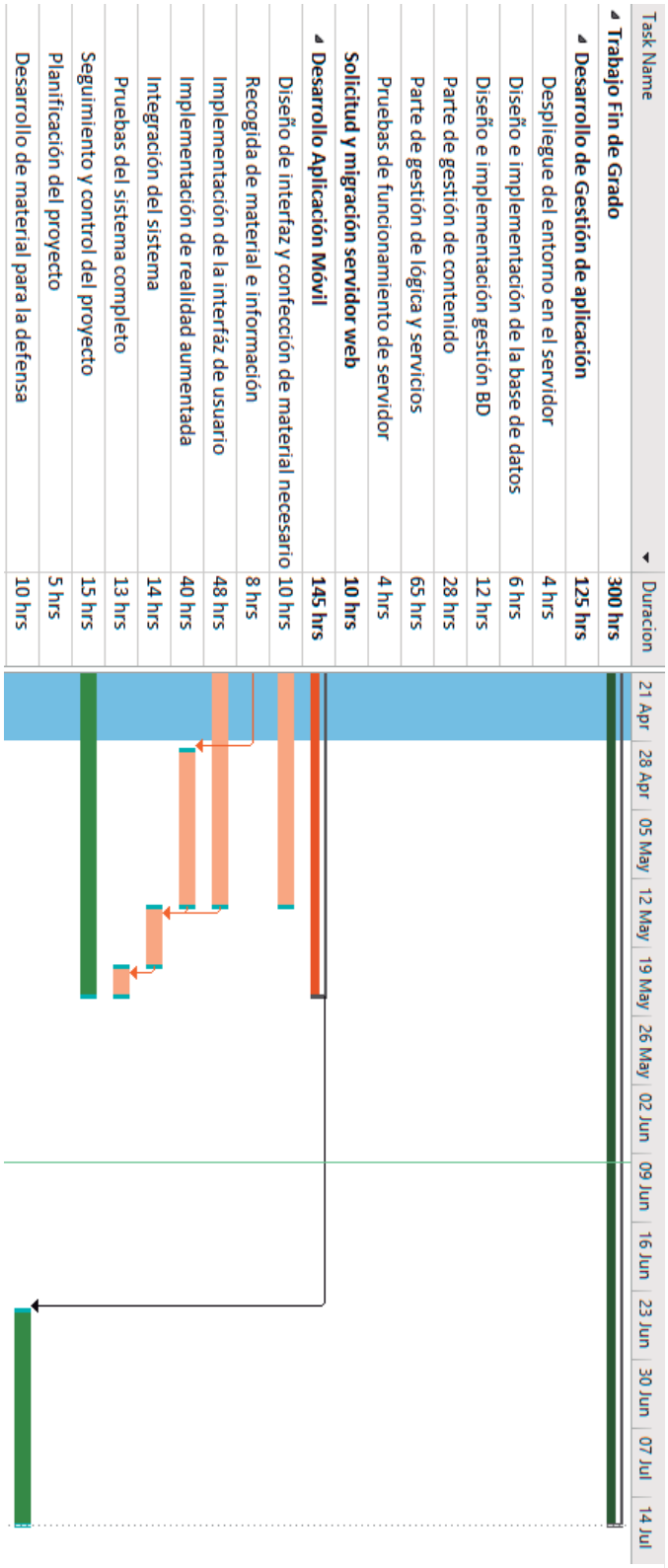


Fig. 16: Diagrama de Gannt 2/2

CONCLUSIONES

RESULTADO FINAL

Una vez finalizado el proyecto dentro del tiempo establecido podemos concluir que se han cumplido todos los objetivos fijados para su correcta elaboración. El producto final cumple con los requisitos que se habían expuesto en un principio y es completamente funcional como hemos podido ver en la fase de pruebas.

Junto a éste documento en formato digital se incluirán todos los productos generados e a lo largo del desarrollo del proyecto, siendo estos:

- Scripts SQL utilizados
- Librerías de gestión de datos del servidor
- Interfaces y librerías de lógica de sistema de servidor y de cliente
- Aplicación de servicios desplegable
- Librería cliente de servicios web
- API para desarrollar interfaces de usuario en Unity
- Proyecto Xcode para iOS
- Aplicación beta para Android

Como hemos mencionado esta última no cumple todos los requisitos fijados para el proyecto, ya que el proyecto se concretó para el desarrollo de la aplicación iOS.

Podremos ver también en el **Anexo II: Implementación - Documentación** la documentación necesaria de cada una de las librerías generadas en el proyecto, ya que no se incluye el código fuente del mismo.

Cabe destacar como uno de los resultados más innovadores el hecho de poder ver información turística en realidad aumentada sobre un monumento. Una vez terminada la aplicación y probada en vivo sobre la ciudad de Logroño queda patente el potencial de la aplicación de estas tecnologías al turismo. Hace fácil y amena la visita de emplazamientos turísticos que las personas que visitan la ciudad pueden sin esta obviar y no tener en cuenta en sus visitas. La combinación del geoposicionamiento con todo esto, hace que con esta aplicación se puede visitar la ciudad completa sin recurrir a guías, oficinas de turismo, etc. Esta aplicación integra todo lo necesario para realizar una visita completa a la ciudad, nos muestra dónde están, qué son y qué importancia tienen cada uno de sus lugares turísticos, además cuando llegamos a cada uno de ellos solo tenemos que abrir la cámara para ver explicaciones sobre el lugar, fotos antiguas, videos relacionados con él, referencias históricas y mucho más y todo ello como si estuviese en el mundo real.

ASPECTOS LEGALES

1. LEY ORGÁNICA DE PROTECCIÓN DE DATOS

Como se ha podido observar en los apartados anteriores de desarrollo, la aplicación recoge datos de los usuarios, incluyendo fotos, etc. Por tanto de acuerdo con la legislatura vigente respecto a la **"Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal"** es necesario en caso de distribución y puesta en servicio de la aplicación realizar un documento de seguridad detallando el tratamiento de los datos que se recogerán, qué seguridad se les aplicará y el nombre de los responsables de ellos, podemos ver una plantilla de este en el ANEXO X.

Una vez realizado este documento, será necesario el registro en la AEPD (Agencia Española de Protección de Datos) de la aplicación con nuestro documento de seguridad detallando todo lo anterior junto con los tipos de datos que se van a tratar que en nuestro caso son de nivel básico. También se debe redactar un acuerdo de privacidad notificando a los usuarios los datos que se van a recoger, con qué fin y durante cuánto tiempo se van a guardar. Los usuarios una vez en conocimiento de esto tendrán que aceptar estos términos para el uso de la aplicación.

Terminado este proceso nuestra aplicación ya estaría dispuesta a ser distribuida cumpliendo con todos los requisitos legales que presenta el estado Español.

2. LICENCIA DE DISTRIBUCIÓN

Todo el software generado, así como el contenido asociado e incluido en el proyecto queda bajo una licencia de tipo "freeware" mediante la cual se permite la distribución, instalación, uso y copia del software completo, o módulos incluidos en el proyecto. No se permite en cambio la modificación o acceso al código fuente de la aplicación quedando restringido el acceso al código fuente al autor del software (Álvaro Pérez Sala).

Como consecuencia de esto no se distribuirá en la presentación del proyecto el código fuente de la aplicación. Si se mostrarán en el **Anexo II: Implementación** partes relevantes del código y que tienen una importancia destacada dentro del marco de desarrollo del proyecto.

MEJORAS

Teniendo en cuenta el producto generado vamos a exponer algunas mejoras que podrían proponerse para continuar el desarrollo de la aplicación en un futuro. En puntos anteriores ya hemos visto algunas de ellas pero recogeremos todas a continuación:

- Implementar casos de uso restantes de la interfaz de usuario. Como especificamos en el alcance del proyecto sería necesaria la limitación de la funcionalidad de la interfaz de usuario para no sobrepasar el tiempo disponible para el proyecto. Por tanto una mejora seria completar el desarrollo de estos casos de uso ya que si están implementados en las capas subyacentes.

- Optimización del funcionamiento del software en dispositivos Android. El objetivo del proyecto consistía en el desarrollo de una aplicación para dispositivos iOS tal y como se ha realizado. Dado que las tecnologías de desarrollo que hemos usado permitían generar a partir del mismo proyecto la aplicación Android se ha generado a modo de prueba no funcional. Para que esta fuera completamente funcional deberíamos optimizar el software para Android igual que hemos hecho para iOS ya que a pesar de tener la mayoría de características funcionando aquellas que son más dependientes del sistema presentan fallos. Se propone esto como una mejora a la aplicación que además dispondría de una base consolidada siendo necesarias solo pequeñas modificaciones en el manejo de llamadas al sistema operativo

- Aumento del contenido de la aplicación. Hoy en día la aplicación se centra en Logroño y concretamente en tres lugares de la ciudad tal y como se especifica en el alcance del proyecto. Con lo cual un proyecto de mejora que podríamos emprender sería el de extender la zona de uso de la aplicación a toda la ciudad y más tarde a otras ciudades.

- Adicción de más características importantes en este ámbito. Sería interesante para una versión futura de la aplicación añadir funcionalidad a la aplicación como por ejemplo la creación de rutas turísticas que los usuarios puedan compartir, la creación de eventos turísticos en la ciudad, tener disponibles notificaciones que nos avisen cuando estamos cerca de cualquier lugar de interés, etc.

- En caso de despliegue registro en la AEPD. Si se llegase a distribuir la aplicación una mejora necesaria sería su registro en la AEPD para cumplir con la legislación recogida en la LOPD, así como todos los trámites que esto conlleva.

LECCIONES APRENDIDAS

Dado el entorno en el que se ha desarrollado el proyecto hemos aprendido mucho sobre la aplicación de tecnologías innovadoras a sectores cotidianos como son el turismo.

Al comienzo del proyecto se hizo la elección de Unity para realizar la parte de aplicación móvil. Esto ha supuesto una ventaja grande de cara a nuestro proyecto ya que permite el desarrollo para Android e iOS y está perfectamente integrado con la librería de realidad aumentada "Vuforia". Con lo cual será el primer entorno de desarrollo en futuros proyectos que tengan realidad aumentada como protagonista.

Por otro lado se intentará prescindir de Unity en otro tipo de proyectos que requieran GUIs clásicas ya que Unity a pesar de ser versátil para desarrollar para varias plataformas no está enfocado a este tipo de aplicaciones y es costoso el desarrollo una GUI clásica sobre él. En nuestro caso las facilidades para el desarrollo de contenido en realidad aumentada compensaba esta cadencia.

En futuros proyectos también se tendrán en cuenta tecnologías como TTS(Text to speech) usada en una escena de realidad aumentada, ya que aportan un valor añadido a la aplicación frente a personas con discapacidad, etc.

Respecto a los servicios web que podamos tener que desplegar en futuros proyectos se intentarán buscar alternativas a SOAP como son JSON o similares, ya que SOAP carga la conexión de muchos datos que con estas tecnologías nos podríamos evitar. Si disponemos de una mala conexión podemos apreciarlo en el funcionamiento de la aplicación.

En cuanto a la metodología de desarrollo, es recomendable el desarrollo modular que se ha llevado a cabo en un proyecto como este que tiene bastantes módulos distintos que se comunican entre sí. A pesar de ello se ha echado en falta el uso de metodologías ágiles en la parte de desarrollo de la aplicación móvil. Con lo cual en un futuro se contemplará esta opción.

BIBLIOGRAFIA

- **Microsoft Developer Portal -** http://msdn.microsoft.com/es-ES/
- **Vuforia Developer Portal -** https://developer.vuforia.com/
- **Unity documentation -** http://docs.unity3d.com/
- **Repositorio de trabajos de fin de grado de la biblioteca de la Universidad de La Rioja -** http://biblioteca.unirioja.es/
- **La Rioja Turismo, Logroño -** http://www.lariojaturismo.com/logrono/

AGRADECIMIENTOS

En primer lugar agradecer a toda mi familia, compañeros y amigos, en especial a Edu, Victor y sobre todo a María, el apoyarme estos meses y haberme hecho más llevadero el trabajo.

A Arturo Jaime Elizondo por dirigir este trabajo y aconsejarme en todas las dudas que me han podido surgir.

Al equipo de CreativiTIC en especial Aratz Setién, Jorge R. López, y Enara Artetxe que me metieron junto con mi compañero Javier Villoslada en éste “mundo” de realidad aumentada y me han apoyado a lo largo de todo el proyecto.

Finalmente a mis padres que han hecho posible este proyecto y llegar hasta aquí.

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO – ANEXO I: DISEÑO

LOGRAR: APLICACIÓN MOVIL PARA EL TURISMO EN LA CIUDAD DE LOGROÑO CON TECNOLOGÍAS DE REALIDAD AUMENTADA

Autor: Alvaro Pérez Sala

Tutor: Arturo Jaime Elizondo

GRADO EN INGENIARÍA INFORMÁTICA

DEPARTAMENTO DE MATEÁTICAS Y COMPUTACIÓN

FACULTAD DE CIENCIAS ESTUDIOS AGROALIMENTARIOS E INFORMÁTICA

1. BASE DE DATOS

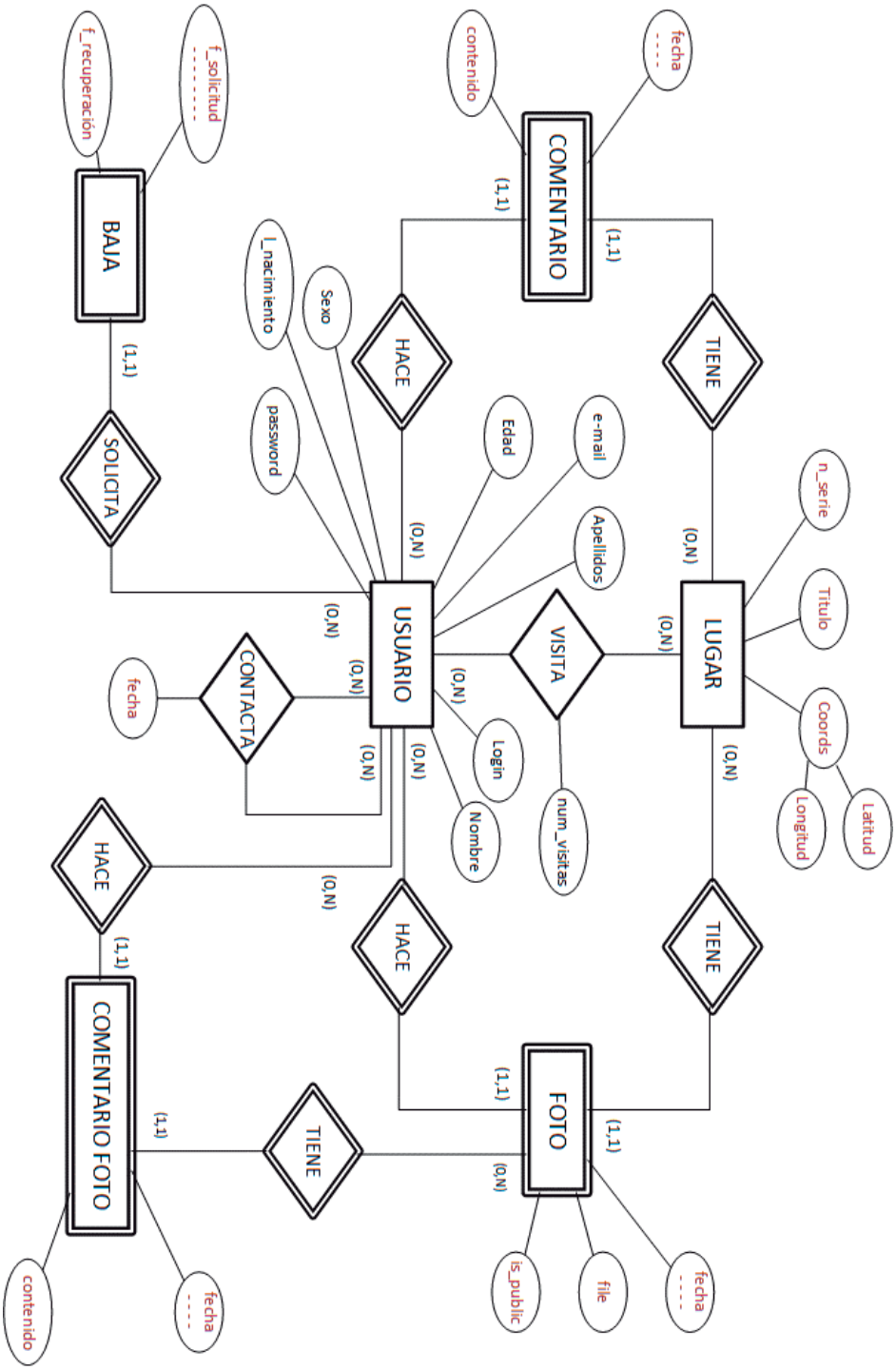


Fig. 1: Diagrama E/R del diseño de la base de datos

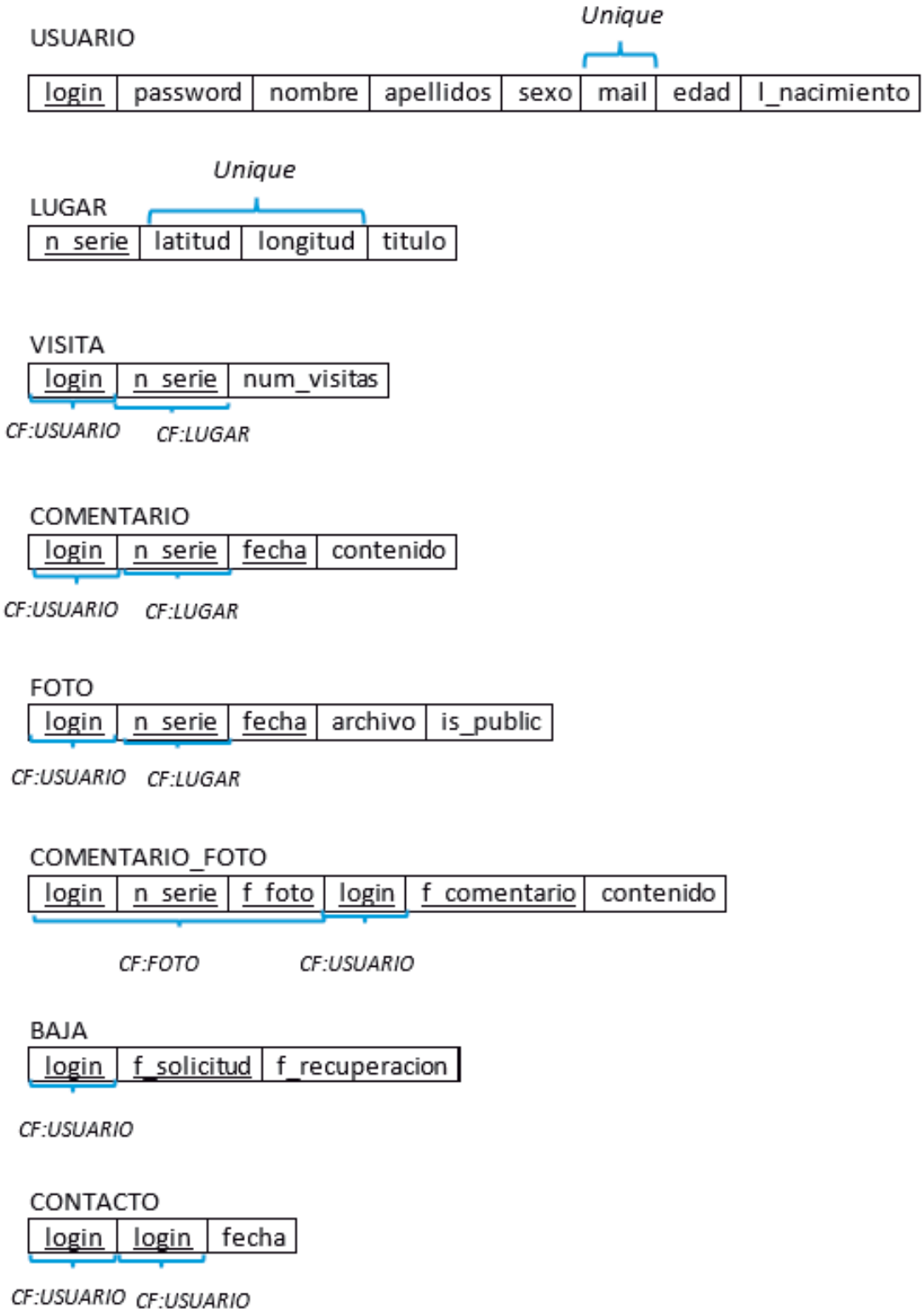


Fig. 2: Modelo relacional del diseño de la base de datos

2. DIAGRAMAS DE PAQUETES Y CLASES

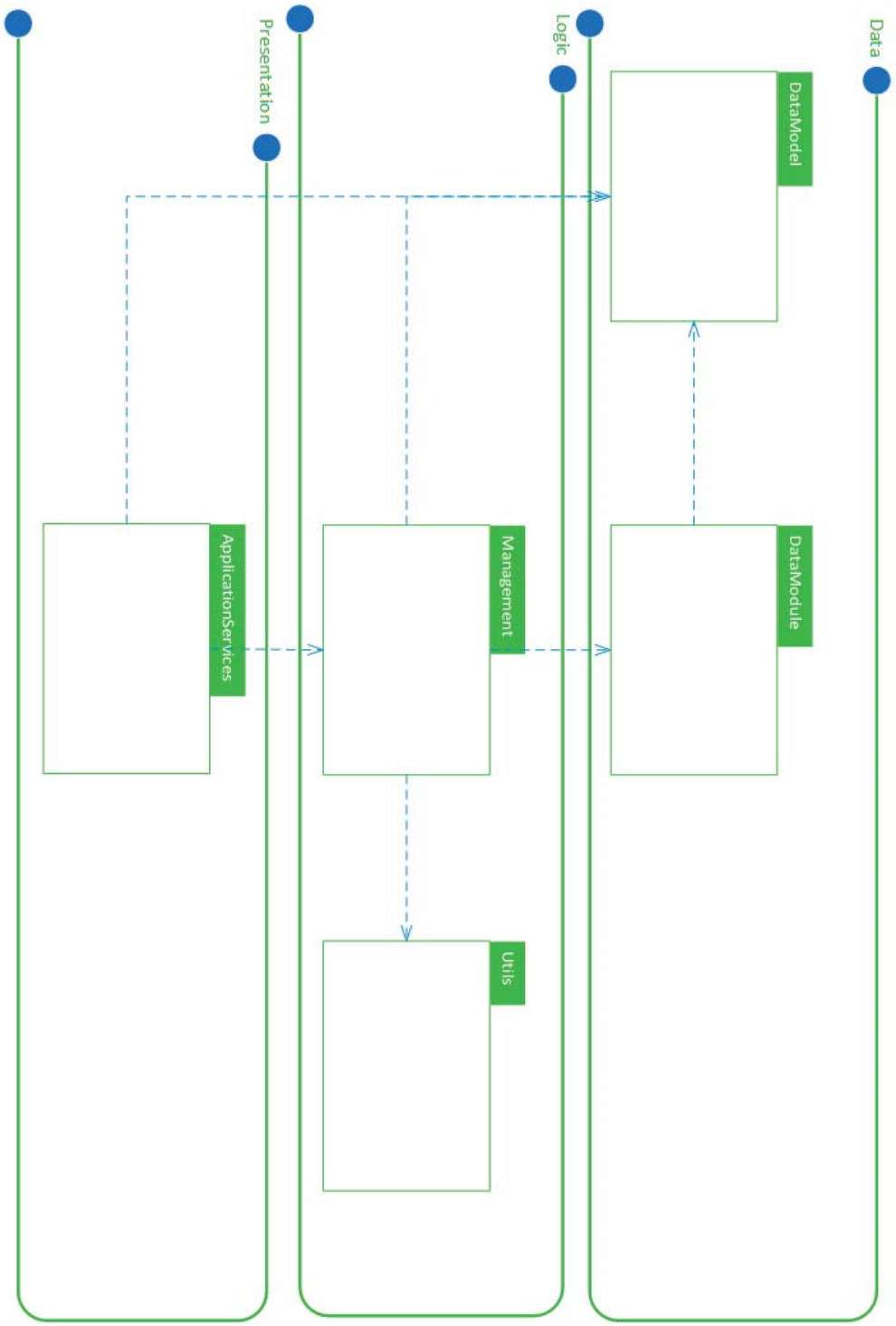


Fig. 3: Diagrama de paquetes de la aplicación de gestión alojada en el servidor

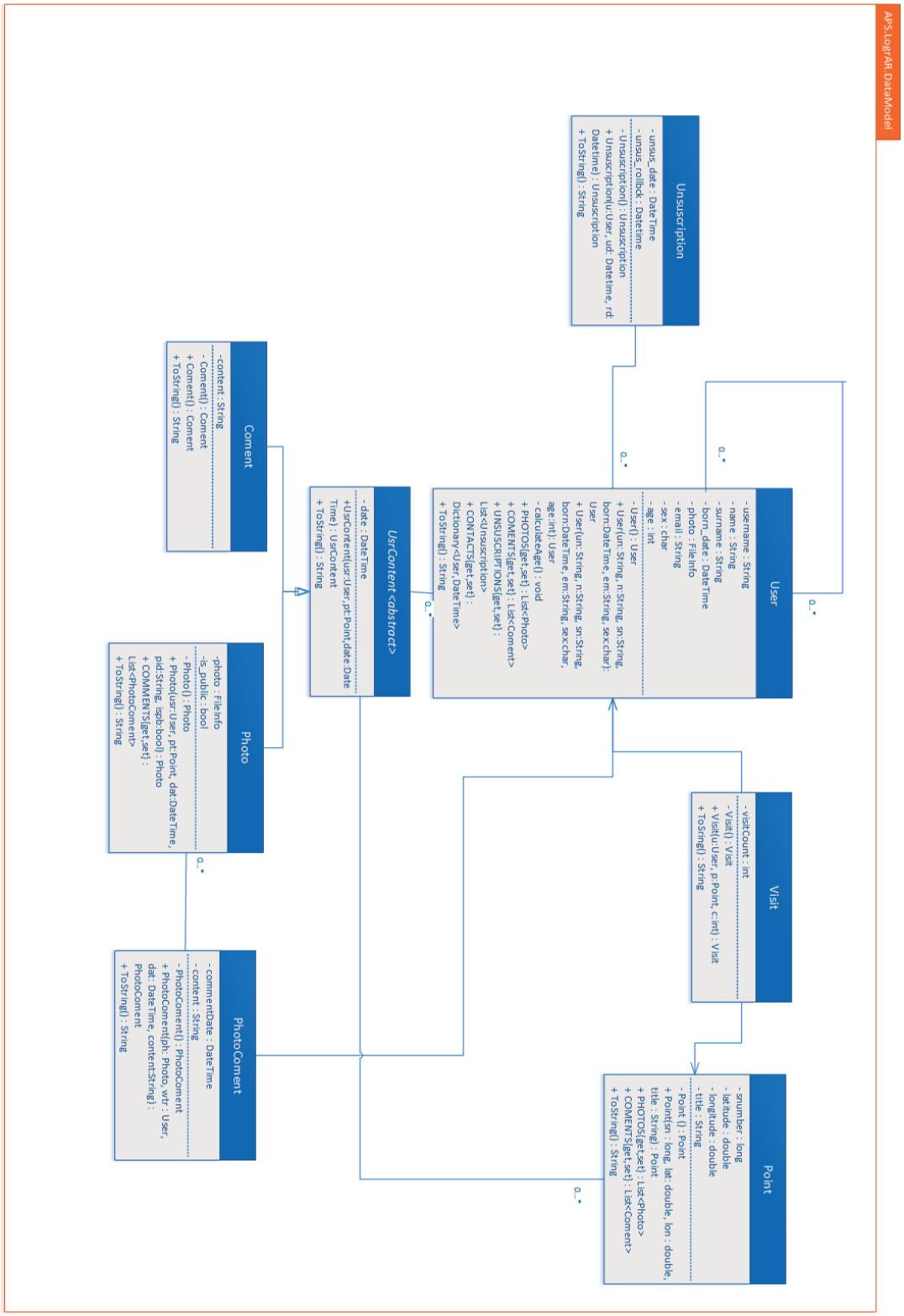


Fig. 4: Diagrama de clases del modelo de datos de la aplicación

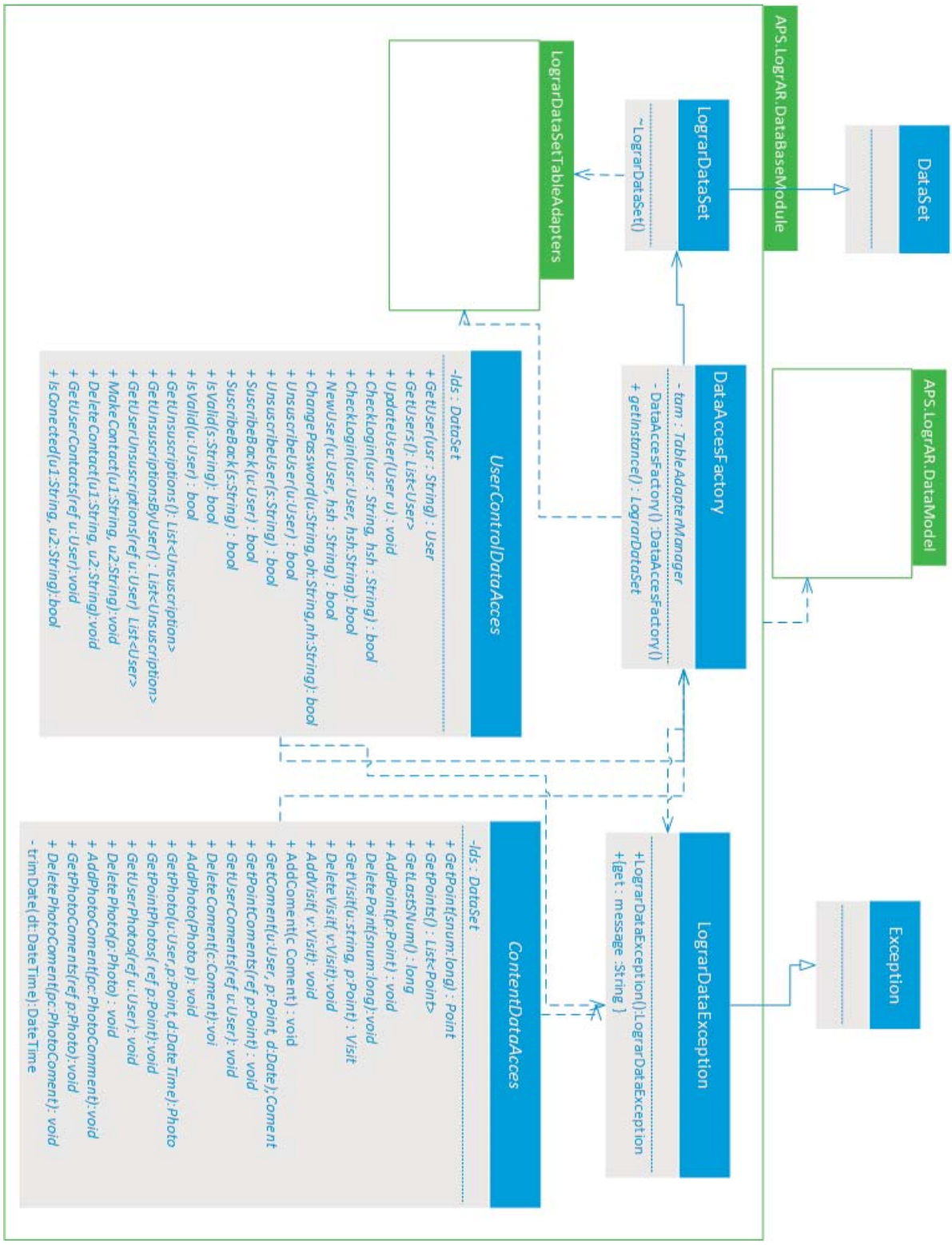


Fig. 5: Diagrama de clases del paquete APS.LogrAR.DataBaseModule

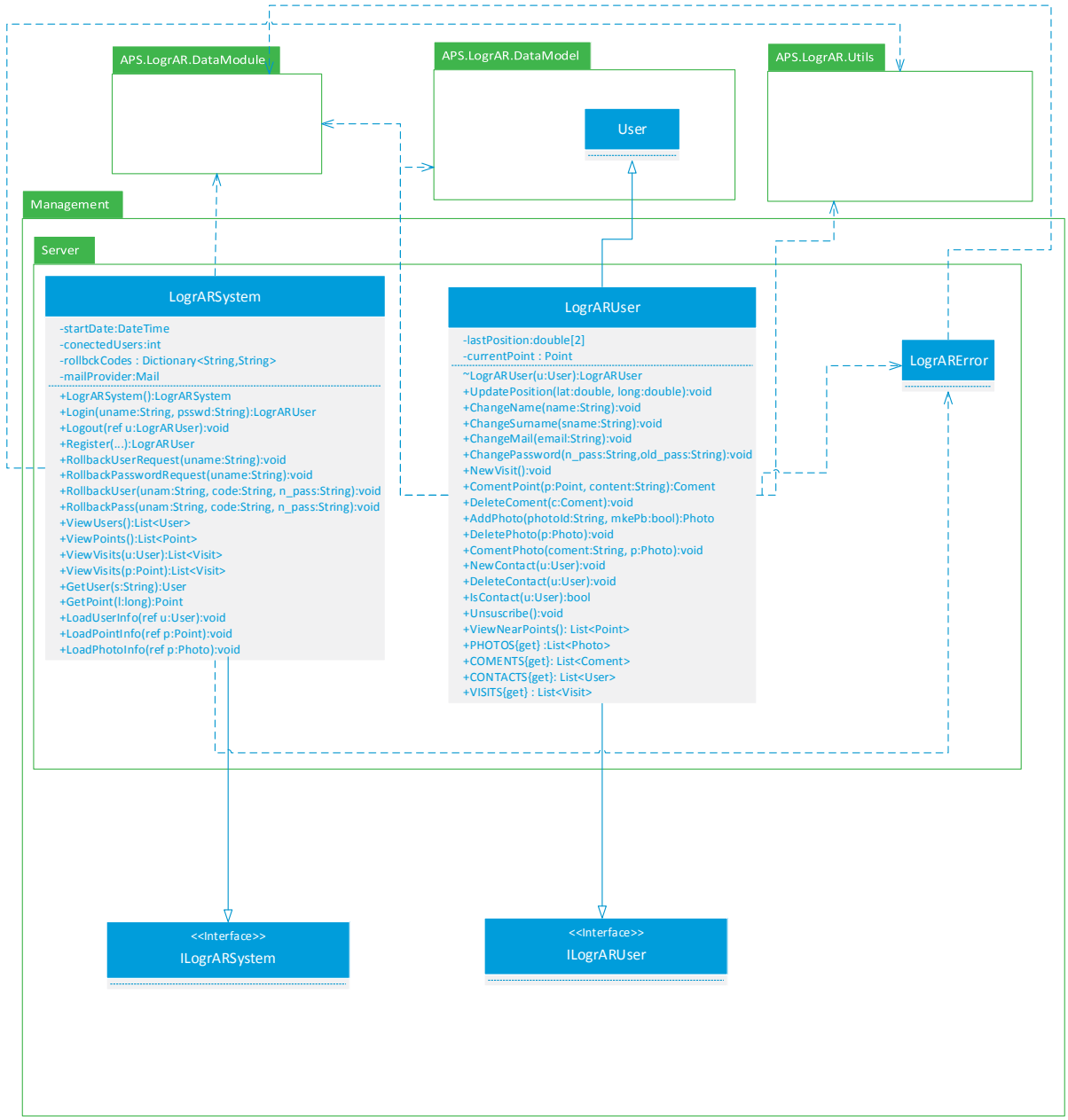


Fig. 6: Diagrama de clases del paquete APS.LogrAR.Management

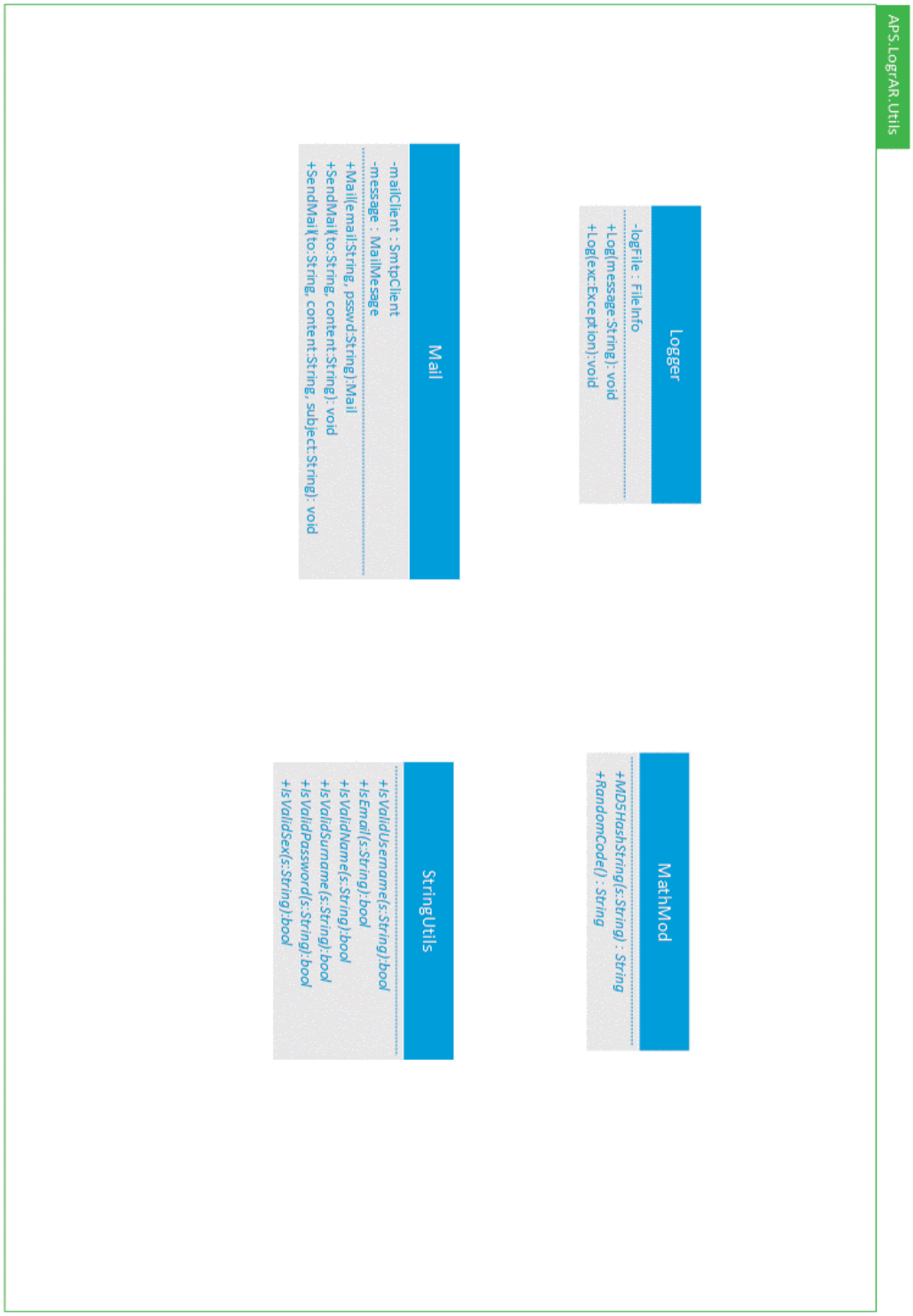


Fig. 7: Diagrama de clases del paquete APS.LogrAR.Utils

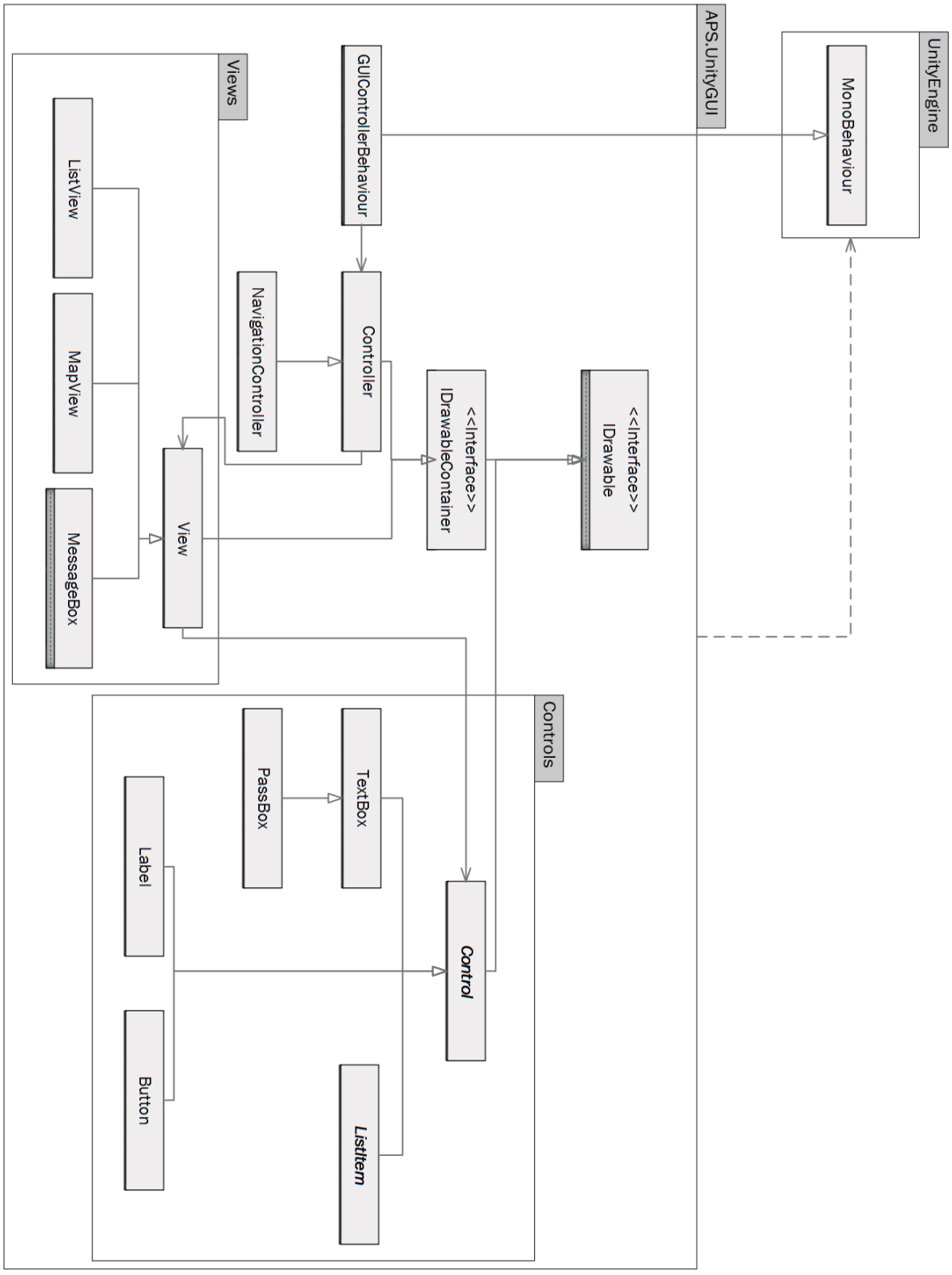
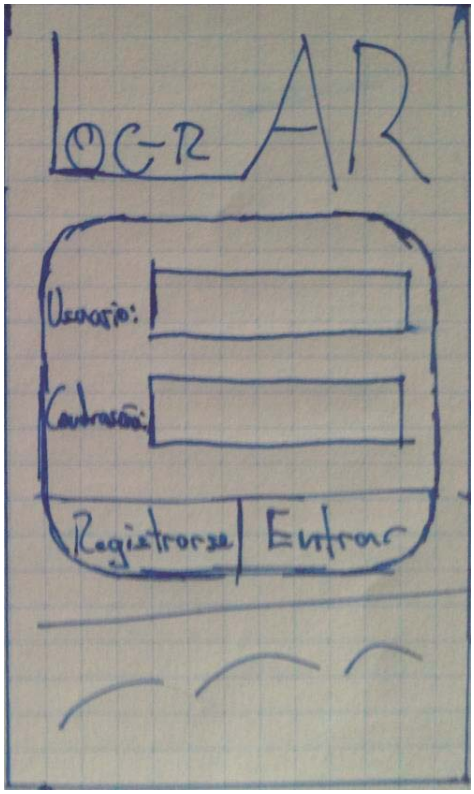
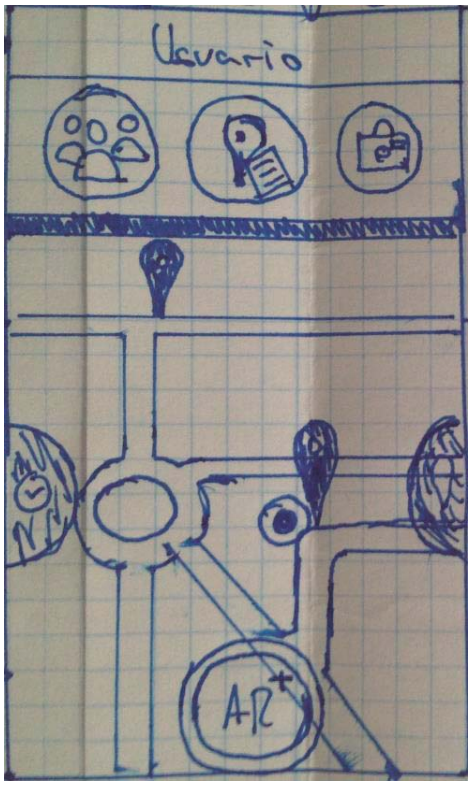
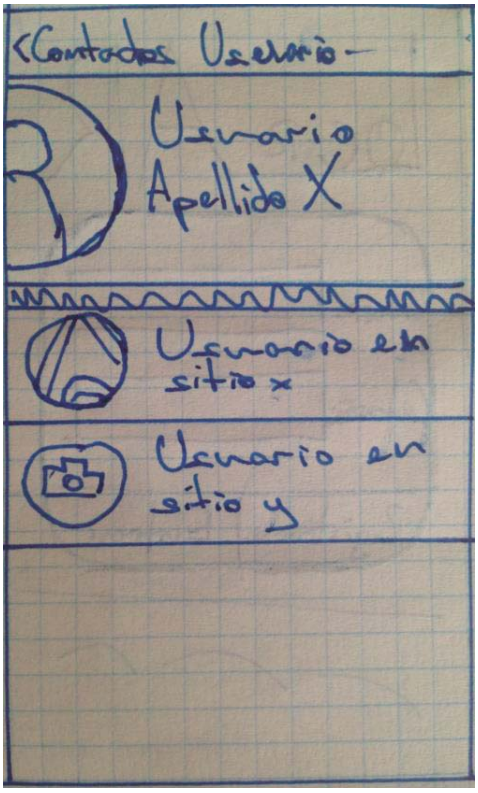


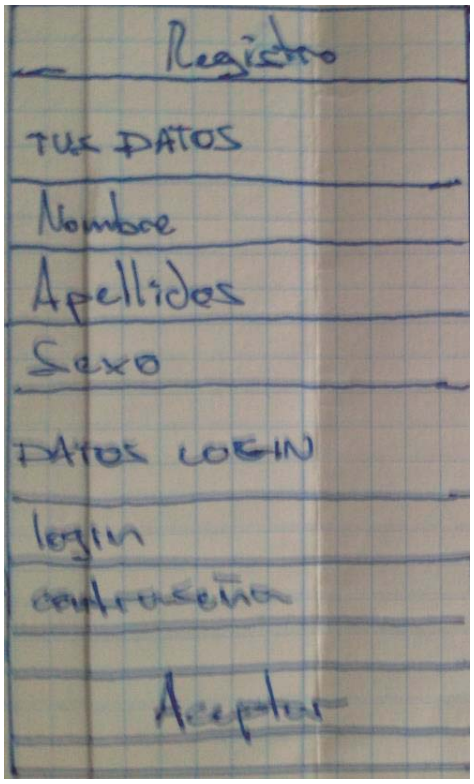
Fig. 8: Diagrama de clases del paquete APS.UnityGUI

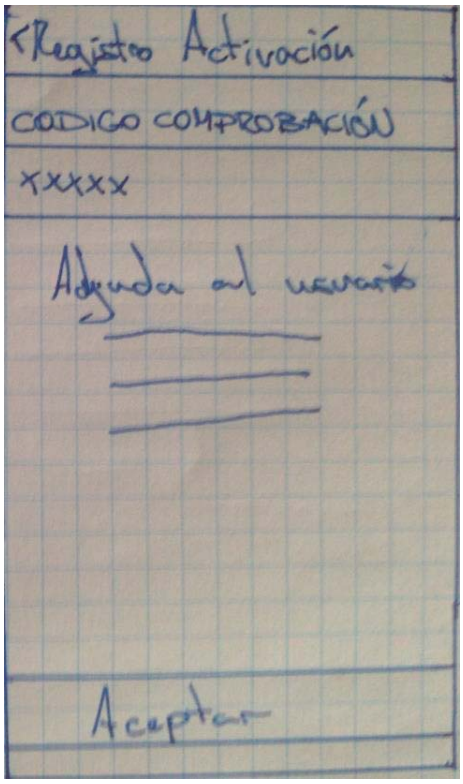
3. DISEÑO VISTAS DE LA APLICACIÓN

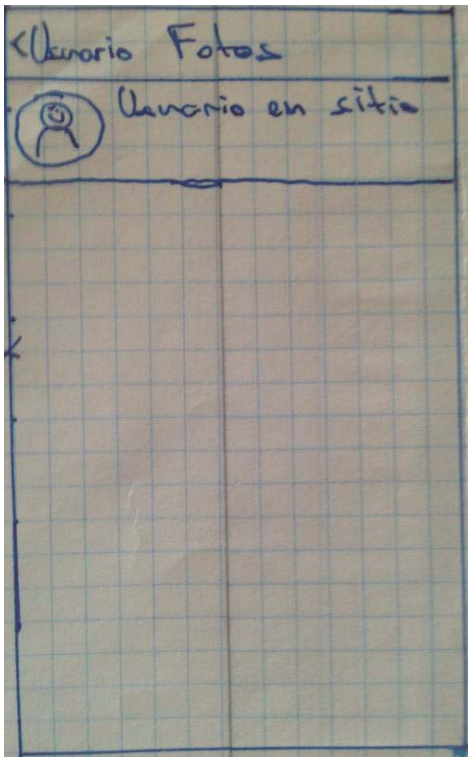




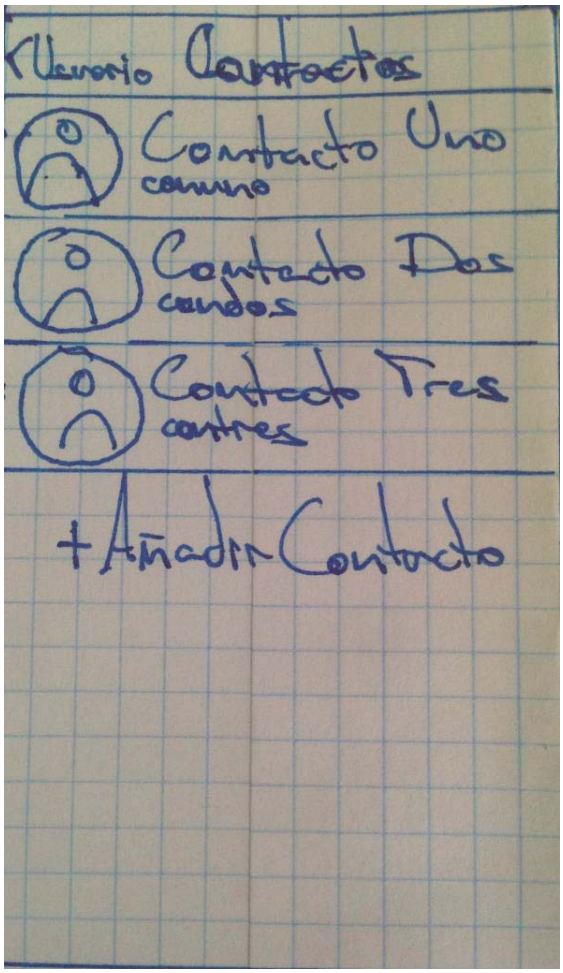


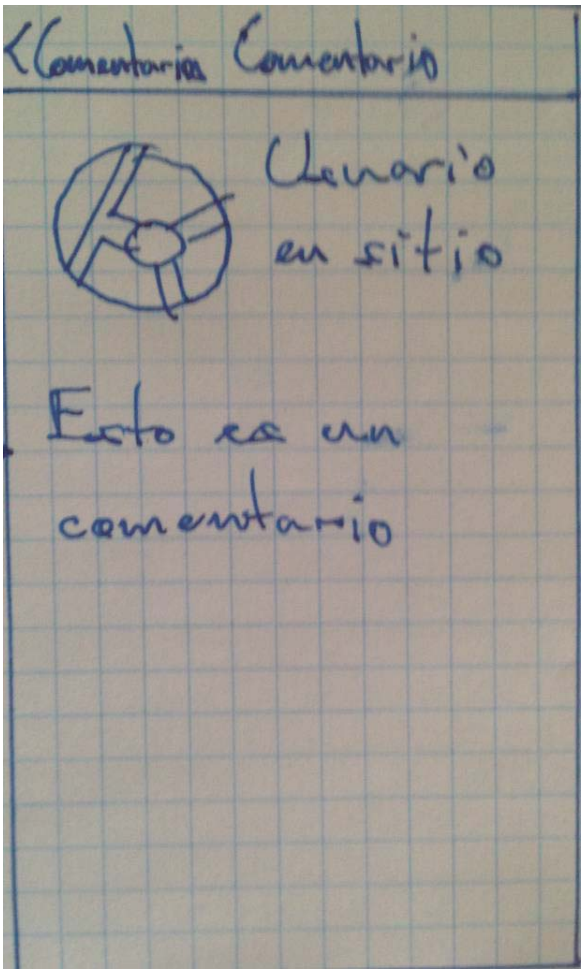

Registro
TUS DATOS
Nombre
Apellidos
Sexo
DATOS LOGIN
login
contraseña
Aceptar


Registro Activación
CODIGO COMPROBACIÓN
xxxxx
Ayuda al usuario
Aceptar


Usuario Fotos
Usuario en sitio


Fotos Foto
Usuario en sitio


Usuario Contactos
Contacto Uno
comun
Contacto Dos
conDos
Contacto Tres
contres
+Añadir Contacto


Comentario Comentario
Usuario
en sitio
Esto es un
comentario

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO – ANEXO II: IMPLEMENTACIÓN

LOGRAR: APLICACIÓN MOVIL PARA EL TURISMO EN LA CIUDAD DE LOGROÑO CON TECNOLOGÍAS DE REALIDAD AUMENTADA

Autor: Alvaro Pérez Sala

Tutor: Arturo Jaime Elizondo

GRADO EN INGENIARÍA INFORMÁTICA

DEPARTAMENTO DE MATEÁTICAS Y COMPUTACIÓN

FACULTAD DE CIENCIAS ESTUDIOS AGROALIMENTARIOS E INFORMÁTICA

1. BASE DE DATOS

SCRIPT DE CREACIÓN

```
CREATE DATABASE [lograr]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'lograr', FILENAME = N'E:\SQLS
DATA\MSSQL11.MSSQLSERVER\MSSQL\DATA\lograr.mdf' , SIZE = 5120KB ,
FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'lograr_log', FILENAME = N'E:\SQLS
DATA\MSSQL11.MSSQLSERVER\MSSQL\DATA\lograr_log.ldf' , SIZE =
1024KB , FILEGROWTH = 10%)
GO
ALTER DATABASE [lograr] SET COMPATIBILITY_LEVEL = 110
GO
ALTER DATABASE [lograr] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [lograr] SET ANSI_NULLS OFF
GO
ALTER DATABASE [lograr] SET ANSI_PADDING OFF
GO
ALTER DATABASE [lograr] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [lograr] SET ARITHABORT OFF
GO
ALTER DATABASE [lograr] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [lograr] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [lograr] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [lograr] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [lograr] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [lograr] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [lograr] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [lograr] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [lograr] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [lograr] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [lograr] SET  DISABLE_BROKER
GO
ALTER DATABASE [lograr] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [lograr] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
```

```
ALTER DATABASE [lograr] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [lograr] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [lograr] SET  READ_WRITE
GO
ALTER DATABASE [lograr] SET RECOVERY FULL
GO
ALTER DATABASE [lograr] SET  MULTI_USER
GO
ALTER DATABASE [lograr] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [lograr] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
USE [lograr]
GO
IF NOT EXISTS (SELECT name FROM sys.filegroups WHERE is_default=1
AND name = N'PRIMARY') ALTER DATABASE [lograr] MODIFY FILEGROUP
[PRIMARY] DEFAULT
GO

USE lograr

CREATE TABLE USERS(
 usrlogin varchar(10) PRIMARY KEY,
 name varchar(24),
 surname varchar(64),
 sex char(1),
 email varchar(32) UNIQUE,
 age int,
 born_date DATETIME)

CREATE TABLE PASSWD(
 usrlogin varchar(10),
 passwd varchar(32),
 CONSTRAINT PK_PASSWD PRIMARY KEY (USRLOGIN),
 CONSTRAINT FK_PASSWD_USERS FOREIGN KEY (USRLOGIN) REFERENCES
USERS(USRLOGIN))

CREATE TABLE POINTS(
 snumber bigint PRIMARY KEY,
 latitude float,
 longitude float,
 title varchar(32) NOT NULL,
 CONSTRAINT CK_POINTS UNIQUE (latitude,longitude))

CREATE TABLE VISITS(
 usr varchar(10),
 point bigint,
 visitCount int,
 CONSTRAINT PK_VISITA PRIMARY KEY (usr,point),
 CONSTRAINT FK_USER_VISITS FOREIGN KEY (usr) REFERENCES
USERS(usrlogin),
 CONSTRAINT FK_POINTS_VISITS FOREIGN KEY (point) REFERENCES
POINTS(snumber))
```

```
CREATE TABLE COMMENTS(
 usr varchar(10),
 point bigint,
 c_date DATETIME,
 content varchar(140),
 CONSTRAINT PK_COMMENTS PRIMARY KEY (usr,point,c_date),
 CONSTRAINT FK_USER_COMMENTS FOREIGN KEY (usr) REFERENCES
USERS(usrlogin),
 CONSTRAINT FK_POINTS_COMENTS FOREIGN KEY (point) REFERENCES
POINTS(snumber))

CREATE TABLE PHOTO(
 usr varchar(10),
 point bigint,
 t_date datetime,
 res varchar(128),
 is_public bit,
 CONSTRAINT PK_PHOTO PRIMARY KEY (USR,POINT,T_DATE),
 CONSTRAINT FK_USER_PHOTO FOREIGN KEY (USR) REFERENCES
USERS(USRLOGIN),
 CONSTRAINT FK_POINTS_PHOTO FOREIGN KEY (POINT) REFERENCES
POINTS(SNUMBER))

CREATE TABLE PHOTO_COMMENT(
 usr varchar(10),
 point bigint,
 p_date datetime,
 writer varchar(10),
 c_date datetime,
 content varchar(140),
 CONSTRAINT PK_PHOTOCOMMENT PRIMARY KEY
(USR,POINT,P_DATE,WRITER,C_DATE),
 CONSTRAINT FK_PHOTO_PHOTOCOMMENT FOREIGN KEY (USR,POINT,P_DATE)
REFERENCES PHOTO(USR,POINT,T_DATE),
 CONSTRAINT FK_USERS_PHOTOCOMMENT FOREIGN KEY (WRITER) REFERENCES
USERS(USRLOGIN))

CREATE TABLE UNSUSCRIBE(
 usr varchar(10),
 unsus_date datetime,
 rollbck_date datetime,
 CONSTRAINT PK_UNSUSCRIBE PRIMARY KEY (USR, UNSUS_DATE),
 CONSTRAINT FK_USERS_UNSUSCRIBE FOREIGN KEY (USR) REFERENCES
USERS(USRLOGIN))

CREATE TABLE CONTACT(
 usr varchar(10),
 friend varchar(10),
 dat datetime,
 CONSTRAINT PK_CONTACT PRIMARY KEY (USR,FRIEND),
 CONSTRAINT FK_USERS_CONTACT1 FOREIGN KEY (USR) REFERENCES
USERS(USRLOGIN),
 CONSTRAINT FK_USERS_CONTACT2 FOREIGN KEY (FRIEND) REFERENCES
USERS(USRLOGIN))
```

PROCEDIMIENTOS ALMACENADOS

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:         Álvaro Pérez Sala
-- Create date: 13/2/2014
-- Description:    Función para obtener el siguiente número de serie
a introducir
-- =============================================
CREATE FUNCTION dbo.GET_LAST_SN ()
RETURNS BIGINT
AS
BEGIN
 DECLARE @ret bigint
 SELECT @ret= MAX(snumber)
 FROM lograr.dbo.POINTS;
 IF (@ret is null) SET @ret = 0;
 RETURN @ret
END;
GO

CREATE FUNCTION dbo.EXISTS_POSITION (@lat float,@lon float)
RETURNS bit
AS
BEGIN
 DECLARE @ret int
 SELECT @ret= COUNT(snumber)
 FROM lograr.dbo.POINTS AS p
 WHERE p.latitude = @lat AND p.longitude = @lon;
 IF (@ret is null)SET @ret=0;
 RETURN @ret;
END;
GO
```

2. ALGORÍTMOS DETECCIÓN DE TOQUES

```
bool is_tap=true; //Diferencia entre tap y scroll
int previous_touches=0;
Vector2 touch_position;
int clock=0;
int tapCount=0;
double previous_pitch=0;

//Métodos manejo de taps
void catchSingleTap(){
	if(previous_touches == 0 && Input.touchCount == 1 && is_tap){
		previous_touches=1;
		touch_position = new
Vector2(Input.touches[0].position.x,Screen.height-
Input.touches[0].position.y);
	}
	if(previous_touches == 1 && Input.touchCount == 1 && is_tap){
		clock++;
if(System.Math.Sqrt(System.Math.Pow(Input.touches[0].deltaPosition.x
,2)+System.Math.Pow(Input.touches[0].deltaPosition.y,2)) > 20 ||
clock>20){
			is_tap = false;
			clock=0;
			if(Scrolling != null)
Scrolling(Input.touches[0].deltaPosition.x,Input.touches[0].deltaPos
ition.y,touch_position.x,touch_position.y);
		}
	}else if(previous_touches == 1 && Input.touchCount == 0 &&
is_tap){
		clock=0;
		previous_touches=0;
		tapCount++;
	}
 }

void handleTaps(){
	if(tapCount == 1){
		clock++;
		if(clock>8){
			tapCount=0;
			clock=0;
			if(Tap != null)
Tap(touch_position.x,touch_position.y);
		}
	}else if(tapCount == 2){
		tapCount=0;
		clock=0;
		if(DoubleTap != null)
DoubleTap(touch_position.x,touch_position.y);
	}
 }
```

```
       //Captura de scroll
void catchScroll(){
       if(!is_tap){
             if(Input.touchCount != 0 && Scrolling != null)
Scrolling(Input.touches[0].deltaPosition.x,Input.touches[0].deltaPos
ition.y,touch_position.x,touch_position.y);
             else{
                   is_tap=true;
                   previous_touches = 0;
             }
       }
 }
```

3. DOCUMENTACIÓN DE LIBRERÍAS

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO – ANEXO III: MARCADORES

LOGRAR: APLICACIÓN MOVIL PARA EL TURISMO EN LA CIUDAD DE LOGROÑO CON TECNOLOGÍAS DE REALIDAD AUMENTADA

Autor: Alvaro Pérez Sala

Tutor: Arturo Jaime Elizondo

GRADO EN INGENIARÍA INFORMÁTICA

DEPARTAMENTO DE MATEÁTICAS Y COMPUTACIÓN

FACULTAD DE CIENCIAS ESTUDIOS AGROALIMENTARIOS E INFORMÁTICA

Fig. 1: Marcador correspondiente a la escena de realidad aumentada "Estatua de Espartero"

Fig. 2: Marcador correspondiente a la escena de realidad aumentada "Plaza del mercado"

Fig. 3: Marcadorcorrespondiente a la escena de realidad aumentada "Puerta del revellin"