Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Reingeniería de una aplicación móvil de realidad aumentada para localización en interiores

Autor:
D. Diego García Marcos

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Reingeniería de una aplicación móvil de realidad aumentada para localización en interiores

Autor:
D. Diego García Marcos

Tutores:
Dr. Diego R. Llanos Ferraris
D. Iván Carreño Calzada

Agradecimientos

A Diego por darme la oportunidad de realizar este proyecto y guiarme en su desarrollo.

A Iván por explicarme y ayudarme con las cuestiones más técnicas sin importar el momento.

A mis compañeros de MoBiVAP y RDNest, por acogerme en los equipos y ayudarme cuándo los necesité, en especial cuando me tuvieron que instalar *Xtremeloc* en casa.

A todos los miembros de la E.T.S. de Ingeniería Informática, gracias a ellos he podido completar mis estudios académicos.

A mi familia por permitirme estudiar este Grado y apoyarme en todas mis decisiones.

A mis amigos por dejarme desconectar y desahorgarme con ellos durante los momentos más complicados de los estudios del Grado.

Resumen

Este Trabajo Fin de Grado consiste en volver poner en funcionamiento una aplicación de realidad aumentada para dispositivos Android que permite la localización en interiores denominada *Skywalker*. Para ello hace uso de los diferentes sensores con los que actualmente cuentan los dispositivos. Esta aplicación utiliza *Xtremeloc* [1], [2], [3], [4], [5], un sistema de localización en interiores, que cuenta con una API pública para su uso.

Se disponía del código y la documentación, pero la aplicación no funcionaba. Para hacerla funcionar se ha seguido un proceso de reingeniería, durante el cuál se ha añadido una nueva funcionalidad que permite realizar un registro de usuario desde la propia App.

Abstract

This Final Degree Project consists of putting into opperation an augmented reality application for Android devices, named *Skywalker*,that allows indoor location. It makes use of the different sensors that currently have the devices This application uses *Xtremeloc* [1], [2], [3], [4], [5], an indoor location system, which has a public API for its use.

The code and documentation were available, but the App did not work. To make it work, a reengineering process has been followed, during which a new functionality has been added that allows users to register from the App itself.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Actualmente los *Smartphones* ofrecen unas capacidades similares a un ordenador personal, con la diferencia de que siempre están en nuestros bolsillos. Por eso se utilizan en tareas geolocalización, aunque sólo se ha profundizado en ella para espacios exteriores.

En este Trabajo Fin de Grado se propone la utilización de los teléfonos inteligentes, para la localización de personas o elementos en espacios interiores aprovechando su, cada vez mayor, capacidad de cómputo

1.1. Contexto y motivación

Los sistemas de geolocalización tradicionales , como el GPS, tienen falta de precisión en espacios *indoor* y utilizan los clásicos mapas en dos dimensiones.

Este TFG, presenta una solución al inconveniente y a la limitación anteriormente indicados:

- Para resolver la falta de precisión, se utilizará el sistema de localización en interiores *Xtremloc*, el cuál permite localizar elementos sin necesidad de disponer de GPS, basándose en la tecnología *Bluethoot*.
- Se utilizará la realidad aumentada para facilitar la localización de el/los elemento/s que se quieran encontrar.

1.2. XtremeLoc

Aunque se han desarrollado diversos TFG's y artículos sobre este sistema [1] [2] [3] [4] [5], a continuación se explica brevemente su funcionamiento.

XtremeLoc es una aplicación que permite la localización en interiores y que consta de tres elementos:

1.2.1. Emisores

Son dispositivos de bajo consumo y pequeño tamaño que emiten una señal *Bluetooth*. Estas balizas emplean el estándar *iBeacon*, un protocolo desarrollado por Apple.

Entre los datos transmitidos, como puede observarse en la Figura 1.1 [6], se encuentran un *id* universal, un *major* que identifica a un grupo, un *minor* que identifica a un solo elemento, el nivel de batería y la potencia a la que se transmiten los datos. Estos dispositivos, denominados *beacons*, pueden ser configurados mediante aplicaciones móviles, como por ejemplo *iBKS Config Tool*.

El comportamiento de estos *beacons* puede ser simulado por dispositivos móviles, esto permitirá enviar una señal mediante el protocolo *iBeacon* desde nuestro *Smartphone*.

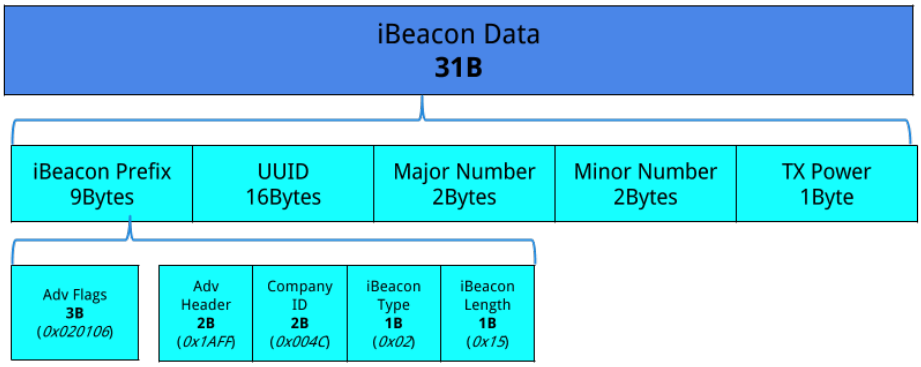


Figura 1.1: Formato de la trama *iBeacon*

1.2.2. Antenas

Las antenas están fabricadas mediante *Raspberry Pi 3*, un modelo que cuenta con *Bluetooth* y *wifi* integrados, una interfaz GPIO y puertos USB. Debido a que la interfaz *wifi* y *Bluetooth* comparten el mismo hardware, las *Raspberry's* llevan conectado un emisor *wifi* externo en uno de los puertos USB, con el objetivo de reducir interferencias en el *Bluetooth*. Su sistema operativo consiste en una distribución *Linux* personalizada basada en *Buildroot* [7] y que cuenta con un sistema de actualizaciones remoto, ya que normalmente las antenas se encuentran situadas en el techo, y así evitar tener que acceder a ellas físicamente.

Las señales emitidas por las balizas son recogidas por estas antenas situadas en diversos puntos de un edificio. Las antenas se encargan de calcular la distancia a la que se encuentran los emisores en función de la intensidad de la señal recibida, después mandan esas distancias al servidor a través de una petición HTTP de tipo POST.

1.2.3. Servidor

El servidor se encarga de calcular mediante un algoritmo de trilateración iterativo, similar a los algoritmos utilizados en los GPS, la posición del emisor a partir de las distancias recibidas. Cuenta con una API REST con la que se pueden comunicar sistemas externos.

1.3. Punto de partida

Se ha partido de la aplicación *Skywalker*, desarrollada por Héctor del Campo en el TFG *Localización de personas y objetos a través de realidad aumentada en sistemas móviles* del año 2017.

1.4. Objetivos

- Poner en funcionamiento la aplicación ya desarrollada, ya que *Xtremeloc* ha cambiado desde que se desarrolló la App y ésta no se ha ido adaptando a dichos cambios.
- Permitir a un nuevo usuario registrarse, ya que en la versión original, sólo se podía acceder con un usuario previamente creado por el administrador de *Xtremeloc*.

1.5. Estructura del documento

La estructura de la memoria será la siguiente:

- **Capítulo dos. Análisis de requisitos.** Se definirán los requisitos que tendrá que satisfacer la aplicación.
- **Capítulo tres. Plan de proyecto.** En el se presentará la planificación del proyecto: sus fases, dependencias y riesgos.
- **Capítulo cuatro. Modelo de análisis.** Se detallarán los diferentes casos de uso del sistema y el modelo de dominio extraído de los requisitos.
- **Capítulo cinco. Diseño.** En el se describirá la arquitectura del sistema, el modelo de dominio y los diagramas de secuencia, así como las diferentes vistas de las que consta la aplicación. Además se explica toda la problemática que aparece a la hora de desarrollar la aplicación: concurrencia, ángulo de visión de la cámara, transmisión de la señal *Bluetooth*...
- **Capítulo seis. Reingeniería.** Se detallará, con la ayuda de fragmentos de código, los diferentes cambios que se han realizado. También se explicará la nueva funcionalidad que se ha incorporado al sistema.
- **Capítulo siete. Implementación.** En él se describirán las soluciones dadas a los problemas descritos en el Capítulo 5.
- **Capítulo ocho. Pruebas.** Se presentarán las pruebas realizadas para comprobar que todos los requisitos del sistema quedan satisfechos.
- **Capítulo nueve. Manual de programador.** En él aparecerá todo lo necesario para que un desarrollador pueda modificar la aplicación o ampliar sus funcionalidades fácilmente.

- **Capítulo diez. Manual de usuario.** Incluirá todo lo necesario para que el usuario sea capaz de utilizar la aplicación y que pueda resolver las dudas que le surjan durante su uso.
- **Capítulo once. Conclusiones y trabajo futuro.** Se expondrán las conclusiones extraídas de la realización del TFG. Además se enumerará brevemente lo que se ha realizado y lo que se podría haber hecho.

Después de esta pequeña introducción, en el siguiente capítulo se abordará el análisis de requisitos del sistema a realizar.

Capítulo 2

Análisis de requisitos

Este capítulo está dedicado al análisis de requisitos del sistema. En él se describen los requisitos funcionales, no funcionales y de información que tiene que satisfacer el sistema que se va a desarrollar.

El objetivo de este capítulo es comprender como tiene que ser el sistema, para poder realizar correctamente las etapas posteriores: planificación del proyecto y modelo de análisis de la aplicación.

2.1. Definición de los Actores

La aplicación móvil sólo interactuará con el usuario final. Dentro de los diferentes roles que puede tener un usuario en *Xtremeloc* (sistema de localización en interiores que utiliza la aplicación), a los que usan esta aplicación se les da el rol de usuario móvil.

2.2. Requisitos Funcionales

A continuación se listan las funcionalidades que debe tener el sistema, se han mantenido las ya presentadas en el TFG base:

RF-01: Conexión externa

El sistema deberá conectarse a un servidor para recibir información sobre los diferentes elementos localizables.

RF-02: Seleccionar elementos

El sistema deberá permitir al usuario seleccionar elementos de su interés.

RF-03: Visualizar elementos

El sistema permitirá al usuario mostrar cualquier elemento de su interés.

RF-04: Actualizar elementos

El sistema actualizará los elementos seleccionados por el usuario, distancia/orientación/piso, si se moviesen dichos elementos o el propio usuario.

RF-05: Ayuda a la orientación

El sistema indicará al usuario hacia dónde ha de dirigir el dispositivo para localizar cada punto de su interés.

RF-06: Visualización cámara trasera

El sistema utilizará la imagen de la cámara trasera como fondo dónde mostrar la información.

RF-07: Cierre de sesión

El sistema permitirá realizar desconectarse del servidor.

Además de los anteriormente citados, *Xtremloc* deberá satisfacer el siguiente:

RF-08: Registrar nuevo usuario

El sistema permitirá al usuario, crear un nuevo usuario para poder realizar la conexión con el servidor. Este nuevo usuario quedará guardado en el servidor para futuras conexiones.

2.3. Requisitos No Funcionales

Las restricciones a las que debe ajustarse el sistema son las mismas que las mencionadas en el TFG de partida:

RNF-01: Disponibilidad Android

El sistema deberá funcionar en dispositivos Android 5.0 o superior, nivel de API 20 y que cuenten con cámara trasera, sensores de rotación y *Bluetooth LE*.

RNF-02: Tipo de conexión

El sistema deberá utilizar una conexión de red, mediante HTTPS y peticiones tipo REST, para comunicarse con el servidor.

RNF-03: Transmisión posición

El sistema deberá transmitir su localización e identificador a antenas receptoras de paquetes *Bluetooth LE*.

2.4. Requisitos de Información

También se han mantenido los requisitos mínimos sobre la información que se debe almacenar en el sistema citados en el TFG original:

RI-01: Posicionamiento

Por cada punto disponible, incluido el propio usuario:

- Identificador
- Nombre
- Posición

RI-02: Centro

Por cada centro:

- Identificador
- Escala del mapa
- Norte magnético

Una vez definidos todos los requisitos del sistema, en el siguiente capítulo se elaborará el plan de proyecto a seguir durante el desarrollo del mismo.

Capítulo 3

Plan de proyecto

En este capítulo se detalla el plan de proyecto que se ha seguido en el desarrollo de de este TFG.

3.1. Resumen del proyecto

3.1.1. Propósito, alcance y objetivos

El objetivo de este TFG es el desarrollo de una aplicación móvil que sea capaz de localizar elementos en espacios interiores en los que esté instalado el sistema *Xtremeloc*, así como poder crear, de manera local, puntos de interés, que posteriormente se puedan querer buscar.

3.1.2. Definiciones y acrónimos

A continuación se enumeran los acrónimos que se emplearán a lo largo del documento.

- URL: Uniform Resource Locator, identificador cuyos recursos referidos pueden cambiar. Se utiliza como sinónimo de dirección web.
- API: Application Programming Interface, conjunto de rutinas que provee acceso a funciones de un determinado software.
- REST: REpresentational State Transfer, estilo de arquitectura software.
- UML: Unified Modeling Language, lenguaje de modelado de sistemas software.
- HTTP: Hypertext Transfer Protoco, protocolo de comunicación utilizado en las páginas web. Si a este protocolo se añade seguridad sus siglas serían HTTPS.
- JSON: JavaScript Object Notation, formato de texto sencillo para el intercambio de datos.

3.1.3. Método escogido para el proyecto

Para la realización del proyecto se ha decidido utilizar el método iterativo, ya que hay dos grupos de tareas bien diferenciados: las encaminadas a hacer funcionar la aplicación de la que se parte y aquéllas que dotarán a la aplicación de la funcionalidad anteriormente descrita.

- En la primera iteración se realizarán los cambios necesarios para que la aplicación funcione, ya que como se expuso con anterioridad, se han realizado modificaciones en el sistema en el que se apoya.
- En la segunda iteración se dotará a la aplicación de nueva funcionalidad: registro de nuevos usuarios.

El método de desarrollo iterativo asegura que cuándo finalicen cada una de las iteraciones se obtendrá un sistema que funciona.

Cada una de estas iteraciones seguirá un desarrollo en cascada ya que tenemos 5 fases bien diferenciadas: análisis, diseño, implementación, pruebas y mantenimiento. Cada fase no comenzará hasta que no haya finalizado la anterior.

Para que la memoria sea más manejable y fácil de comprender, se ha decidido asignar un capítulo a cada una de las fases, en los cuáles se incluirá la información relativa a ambas iteraciones.

3.1.4. Evolución del plan

El plan de desarrollo debe definir cada una de las fases que comprenderán el proyecto y detallar como se gestionará su evolución. Un plan completo, al menos, de estar formado por:

- Definición precisa del producto a elaborar.
- Estimación de los recursos necesarios para completar cada tarea.
- Estimación del tiempo requerido para la elaboración del proyecto.
- Calendarización de las tareas, para analizar el proyecto en el plazo establecido.
- Estudio de los riesgos que tiene el proyecto y método de actuación, si se presentasen.

3.1.5. Ciclo de vida del proyecto

Como se ha mencionado anteriormente, se ha optado por un método iterativo, en el que cada una de las dos iteraciones sigue el método de desarrollo en cascada, en el cuál hasta que no finaliza una de las fases no comenzará la siguiente. A continuación se describen las fases anteriormente enumeradas:

- **Fase de análisis:** esta fase se dedicará a comprender el problema. Como ya existe una versión de la aplicación, se analizará su código y se estudiará la documentación existente. Además, se tendrá que comprender el funcionamiento de *XtremeLoc*.

- **Fase de diseño:** se establecerá la arquitectura del sistema y se definirá el dominio del problema.
- **Fase de implementación:** se desarrollará el *software* siguiendo el método anteriormente descrito.
- **Fase de pruebas y mantenimiento:** se diseñaran y, posteriormente, se ejecutarán una serie de pruebas para comprobar que el sistema satisface los requisitos y no existen fallos. También se elaborarán los manuales de usuario y programador.

3.2. Gestión del proceso

En este apartado se especifican los diferentes planes necesarios para la gestión del proceso.

3.2.1. Plan de puesta en marcha

Se realizará la estimación de cada una de las fases del proyecto teniendo en cuenta la experiencia propia en proyectos similares.

El proyecto se realiza de manera individual por lo que no es necesaria la gestión de personas y organización de grupos.

Para la realización de este proyecto son necesarios los siguientes conocimientos previos:

- Conocimiento en Java y de aplicaciones Android
- Conocimiento en peticiones REST
- Conocimiento en transmisión de paquetes *Bluetooth*
- Conocimiento de la API de *XtremeLoc*

3.2.2. Plan de trabajo

En esta sección se detallan las actividades de las que se compone el proyecto, separadas por fases. Además se incluye una breve descripción, una estimación en días y sus actividades predecesoras, para poder realizar la calendarización de las mismas teniendo en cuenta sus dependencias.

Fase de análisis

Esta primera fase se corresponde con las actividades de análisis del problema. Como se parte de una aplicación ya realizada, se tendrá que comprender para poder estudiar los cambios necesarios.

ID: 01	Inicio de la Fase de Análisis

Tabla 3.1: Actividad 01

ID: 02	Comprensión del proyecto de partida
Predecesoras:	01
Duración:	14 días
Estudio y comprensión del proyecto del que se parte. Esta actividad incluye tanto la documentación como el código fuente.	

Tabla 3.2: Actividad 02

ID: 03	Adquisición de conocimiento en operaciones REST
Predecesoras:	02
Duración:	5 días
Estudio de operaciones REST, poniendo especial interés en las que se usan en el proyecto.	

Tabla 3.3: Actividad 03

ID: 04	Adquisición de conocimiento de la API de XtremLoc
Predecesoras:	03
Duración:	10 días
Estudio de la API de XtremeLoc, prestando especial atención en los métodos que se usan en la aplicación.	

Tabla 3.4: Actividad 04

ID: 05	Análisis de requisitos
Predecesoras:	04
Duración:	5 días
Elaboración de la lista de requisitos funcionales, no funcionales y de información que debe cumplir el sistema.	

Tabla 3.5: Actividad 05

ID: 06	Análisis de riesgos
Predecesoras:	04
Duración:	3 días
Elaboración de la lista de riesgos que se pueden producir durante el desarrollo del proyecto, junto con su plan de contingencia.	

Tabla 3.6: Actividad 06

ID: 07	Modelado de casos de uso
Predecesoras:	05
Duración:	3 días
Descripción de los casos de uso obtenidos a partir de los requisitos junto con su diagrama correspondiente.	

Tabla 3.7: Actividad 07

ID: 08	Modelo de dominio
Predecesoras:	07
Duración:	3 días
Elaboración del modelo de dominio para la fase de análisis.	

Tabla 3.8: Actividad 08

ID: 09	Definición y secuenciación de actividades
Predecesoras:	06, 07
Duración:	3 días
Listado de las actividades a realizar y el orden en el que se llevarán a cabo.	

Tabla 3.9: Actividad 09

ID: 10	Calendarización
Predecesoras:	09
Duración:	2 días
Estimación temporal de cada una de las actividades y definición de sus actividades predecesoras, para posteriormente realizar los correspondientes diagramas de Gantt.	

Tabla 3.10: Actividad 10

ID: 11	Elaboración del plan de proyecto
Predecesoras:	10
Duración:	3 días
Elaboración del plan de proyecto, en el que si incluyen todas las actividades y sus fases, su calendarización y sus riesgos.	

Tabla 3.11: Actividad 11

ID: 12	Fin de la Fase de Análisis

Tabla 3.12: Actividad 12

Fase de diseño

Las tareas que se incluyen en esta fase tienen que ver con el diseño del sistema a implementar. Gran parte consistirá en la elaboración de diagramas explicativos sobre los diferentes aspectos de la estructura del proyecto.

ID: 13	**Inicio de la Fase de Diseño**

Tabla 3.13: Actividad 13

ID: 14	**Elaboración de la arquitectura del sistema**
Predecesoras:	13
Duración:	3 días
Elaboración del diagrama que representa la arquitectura del sistema en el que se incluyen cada una de las capas de las que consta el sistema, con sus correspondientes clases, así cómo las relaciones existentes entre ellas.	

Tabla 3.14: Actividad 14

ID: 15	**Bocetos de interfaz de usuario**
Predecesoras:	13
Duración:	2 días
Creación de los bocetos de las principales vistas de las que consta la interfaz de usuario de la aplicación.	

Tabla 3.15: Actividad 15

ID: 16	**Modelo de dominio**
Predecesoras:	14
Duración:	10 días
Creación del modelo de dominio del sistema.	

Tabla 3.16: Actividad 16

ID: 17	**Diagramas de secuencia**
Predecesoras:	16
Duración:	14 días
Elaboración de los diagramas de secuencia de las principales funcionalidades de la aplicación.	

Tabla 3.17: Actividad 17

ID: 18	Diagrama de operaciones REST
Predecesoras:	17
Duración:	5 días
Creación del diagrama de las operaciones REST que la aplicación utiliza para su correcto funcionamiento.	

Tabla 3.18: Actividad 18

ID: 19	Fin de la Fase de Análisis

Tabla 3.19: Actividad 19

Fase de implementación

En esta fase se generará el código, siguiendo los modelos realizados en la fase anterior, con el objetivo de satisfacer los requisitos establecidos en la fase de análisis.

ID: 20	Inicio de la Fase de Implementación

Tabla 3.20: Actividad 20

ID: 21	Control de versiones
Predecesoras:	20
Duración:	2 días
Se establecerá el sistema de control de versiones en un repositorio ya existente. Se enlazará con Android Studio.	

Tabla 3.21: Actividad 21

ID: 22	Modificación de permisos en Xtremloc
Predecesoras:	21
Duración:	5 días
Modificación del sistema de control de permisos de *Xtremloc*, para poder conocer usuarios móviles la posición de otros elementos.	

Tabla 3.22: Actividad 22

ID: 23	Reingeniería operaciones REST
Predecesoras:	22
Duración:	15 días
Modificación de las operaciones REST utilizadas en la aplicación, ya que la API de *Xtremloc* se ha modificado.	

Tabla 3.23: Actividad 23

ID: 24	Modificación del registro del dispositivo en Xtremeloc
Predecesoras:	23
Duración:	10 días
Modificación del registro del dispositivo como beacon en *Xtremeloc*, ya que no lo reconoce.	

Tabla 3.24: Actividad 24

ID: 25	Modificación de datos para calcular coordenadas
Predecesoras:	24
Duración:	10 días
Modificación de los datos con los que se calculan las coordenadas. En el sistema de partida no se actualizan las distancias.	

Tabla 3.25: Actividad 25

ID: 26	Nueva funcionalidad: Registro nuevo usuario
Predecesoras:	25
Duración:	10 días
Implementación del registro de nuevos usuario, sin necesidad de estar ya dados de alta en *Xtremeloc* para poder usar la aplicación.	

Tabla 3.26: Actividad 26

ID: 27	Fin de la Fase de Análisis

Tabla 3.27: Actividad 27

Fase de pruebas y mantenimiento

Por último, una vez la aplicación funciona, en esta fase se incluyen las tareas finales necesarias para dar por concluido el proyecto.

ID: 28	Inicio de la Fase de Pruebas y Mantenimiento

Tabla 3.28: Actividad 28

ID: 29	Pruebas finales
Predecesoras:	28
Duración:	3 días
Realización de pruebas para comprobar que la aplicación satisface los requisitos establecidos y no existen errores en ella.	

Tabla 3.29: Actividad 29

ID: 30	Manual de programador
Predecesoras:	28
Duración:	2 días
Realización del manual de programador, para que se capaz de modificar o añadir funcionalidades a la aplicación	

Tabla 3.30: Actividad 30

ID: 31	Manual de usuario
Predecesoras:	30
Duración:	2 días
Creación de un manual de usuario, para que pueda utilizar la aplicación y resolver posibles dudas.	

Tabla 3.31: Actividad 31

ID: 32	Fin de la Fase de Pruebas y Mantenimiento

Tabla 3.32: Actividad 32

Diagramas de Gantt

A continuación, mediante diagramas de Gantt, se muestra de una forma más gráfica la planificación del proyecto incluida en las tablas anteriores.

Figura 3.1: Fase de Análisis

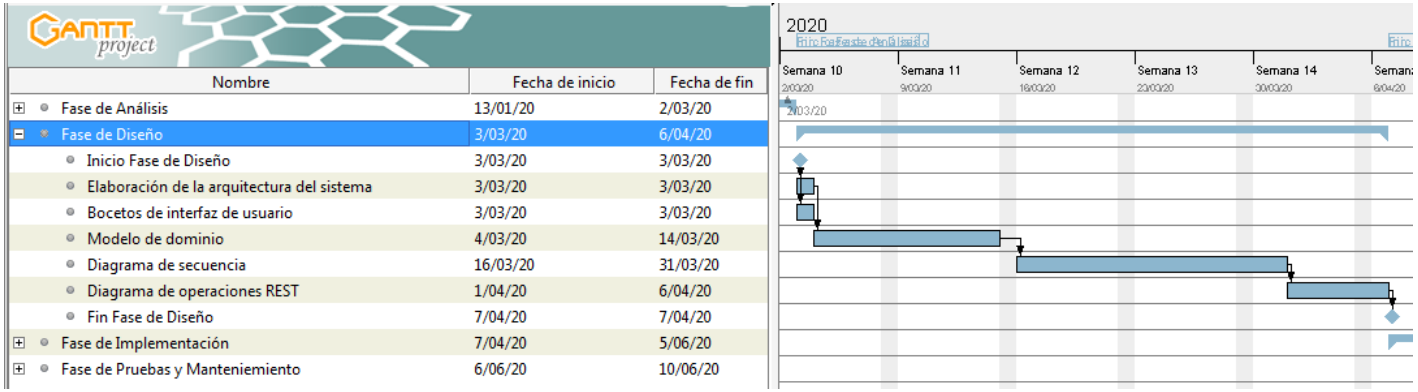
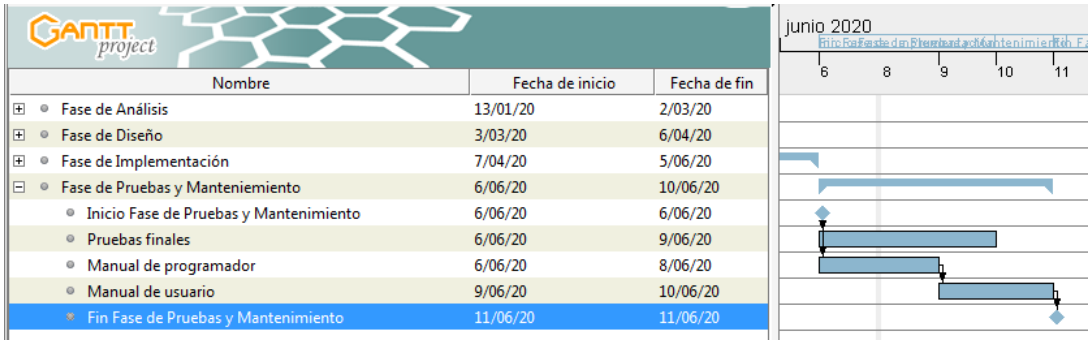


Figura 3.2: Fase de Diseño

Figura 3.3: Fase de Implementación



Nombre	Fecha de inicio	Fecha de fin
Fase de Análisis	13/01/20	2/03/20
Fase de Diseño	3/03/20	6/04/20
Fase de Implementación	7/04/20	5/06/20
Fase de Pruebas y Manteniemiento	6/06/20	10/06/20
Inicio Fase de Pruebas y Mantenimiento	6/06/20	6/06/20
Pruebas finales	6/06/20	9/06/20
Manual de programador	6/06/20	8/06/20
Manual de usuario	9/06/20	10/06/20
Fin Fase de Pruebas y Mantenimiento	11/06/20	11/06/20

Figura 3.4: Fase de Pruebas

3.2.3. Plan de control

En este plan se detallan las mediadas a tomar respecto al control durante el desarrollo del proyecto.

Gestión de Requisitos

Si durante la realización del proyecto los requisitos, una vez ya están definidos, sufren modificaciones se realizarán las siguientes tareas:

1. Analizar y priorizar los cambios.
2. Analizar las implicaciones de realizar dichos cambios.
3. Incorporar los cambios al plan de proyecto.
4. Evaluar los posibles nuevos riesgos debido a estos cambios.
5. Replanificar la calendarización si fuera necesario.

Control de Calendario

Para asegurarse de cumplir con el calendario dispuesto en la planificación, se han establecido hitos para poder detectar posibles desviaciones y así tomar las medidas oportunas para corregirlas. Estos hitos corresponden con la fecha de inicio y fin de cada una de las fases.

3.2.4. Plan de gestión de riesgos

En esta sección se describen los riesgos que pueden surgir durante el desarrollo del proyecto, así como su probabilidad, impacto y plan de actuación.

Mediante la matriz de exposición, que aparece a continuación, se establece la exposición al riesgo teniendo en cuenta su impacto y su probabilidad.

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

Tabla 3.33: Matriz de exposición

R01 - Máquina de desarrollo averiada

La máquina de desarrollo deja de funcionar correctamente.

Impacto: Catastrófico

Probabilidad: Muy baja

Exposición: Baja

Plan de protección: Utilizar un sistema de control de versiones, teniendo así una copia de seguridad en un repositorio remoto.

Plan de contingencia: Se podrá continuar el desarrollo en otra máquina, ya que el trabajo realizado hasta la fecha está en dicho repositorio remoto.

R02 - Dispositivos móviles averiados

El dispositivo móvil de desarrollo también se puede averiar.

Impacto: Crítico

Probabilidad: Baja

Exposición: Baja

Plan de protección: Ninguno

Plan de contingencia: Conseguir un nuevo dispositivo móvil y replanificar, si fuera necesario.

R03 - Pérdida de datos

Se pierde, parte o la totalidad del trabajo realizado, ya sea código o documentación.

Impacto: Catastrófico

Probabilidad: Muy baja

Exposición: Baja

Plan de protección: Utilizar un sistema de control de versiones, teniendo así una copia de seguridad en un repositorio remoto.

Plan de contingencia: Comprobar el trabajo perdido, estimar el tiempo que se tardaría en rehacerlo y modificar el calendario consecuentemente.

R04 - Repositorio corrupto

El repositorio utilizado para el control de versiones se corrompe.

Impacto: Crítico

Probabilidad: Baja

Exposición: Baja

Plan de protección: Utilizar varios repositorios para el control de versiones.

Plan de contingencia: Utilizar los repositorios disponibles y recuperar aquellos corruptos.

R05 - Caída del sistema de apoyo

El sistema en el que se apoya la aplicación deja de de funcionar

Impacto: Crítico

Probabilidad: Baja

Exposición: Baja

Plan de protección: Ninguno

Plan de contingencia: Contactar con el responsable y modificar la planificación si fuese necesario.

R06 - Enfermedad

Debido a una enfermedad se ha de detener el desarrollo del proyecto.

Impacto: Crítico

Probabilidad: Baja

Exposición: Baja

Plan de protección: Ninguno

Plan de contingencia: Evaluar el tiempo perdido y modificar el plan de trabajo.

R07 - Desconocimiento del sistema

Desconocimiento del sistema que se va a modificar.

Impacto: Crítico

Probabilidad: Media

Exposición: Media

Plan de protección: Comprender bien el sistema en la etapa de análisis asignándola el tiempo que sea necesario.

Plan de contingencia: Modificar el calendario en función del tiempo perdido.

R08 - Desconocimiento técnico de los Frameworks

Desconocimiento de las herramientas que se usaran durante la elaboración del proyecto.

Impacto: Crítico

Probabilidad: Media

Exposición: Media

Plan de protección: Asignar el tiempo necesario en la etapa de análisis para comprender todas la herramientas que se usarán durante la realización del proyecto.

Plan de contingencia: Modificar el plan de trabajo, si fuera necesario.

R09 - Lentitud en peticiones REST

Necesidad de una conexión lo suficientemente rápida para poder utilizar peticiones REST.

Impacto: Marginal

Probabilidad: Bajo

Exposición: Bajo

Plan de protección: Mantener las peticiones lo más pequeñas posible.

Plan de contingencia: Cambiar el tipo de conexión con el servidor, modificando el plan de trabajo si fuera necesario.

R10 - Modificación de los requisitos

Los requisitos cambian en una fase posterior a la de análisis

Impacto: Crítico

Probabilidad: Bajo

Exposición: Bajo

Plan de protección: Comprender a la perfección los requisitos en la etapa de análisis.

Plan de contingencia: Actualizar la lista de requisitos y los modelos afectados. Actualizar la planificación, si fuera el caso.

R11 - Mala estimación del tiempo

Alguna de las tareas del camino crítico requieren más tiempo del estimado, por lo que el proyecto finalizara con posterioridad a la fecha prevista.

Impacto: Marginal

Probabilidad: Media

Exposición: Baja

Plan de protección: Identificar perfectamente el camino crítico, comprender cada una de las tareas que lo componen para poder realizar una estimación lo más precisa posible.

Plan de contingencia: Reducir la duración de la tares posteriores, si fuera posible, para que la fecha de finalización del proyecto no se vea afectada.

R12 - Diseño erróneo

El diseño no es correcto

Impacto: Crítico

Probabilidad: Baja

Exposición: Baja

Plan de protección: Realizar el diseño de acuerdo a la fase de análisis y resolver cualquier duda durante la fase de diseño.

Plan de contingencia: Corregir el defecto y modificar el plan de trabajo si fuera necesario.

3.3. Presupuesto

Según la planificación descrita anteriormente, la duración del proyecto será de 5 meses. Teniendo en cuenta que el sueldo medio de un ingeniero informático es de 25.400 euros, y los costes sociales 8.450 euros, se estima que el coste total del proyecto será 14.100 euros.

Debido al confinamiento de la Primavera 2020 y la necesidad del teletrabajo se tuvo que instalar Xtremeloc en el domicilio del autor de este TFG que constó de los siguientes elementos:

- 3 Raspberry-Pi modelo B y sus correspondientes conectoneres a la corriente eléctrica. Coste: 148.07 euros IVA incluido.
- 3 MicroSD GB. Coste: 25,01 euros IVA incluido.

Esto implicó un retraso de dos semanas en la planificación anteriormente descrita, por lo que el coste total aproximado ascendería a 15.700 euros.

Después de describir el plan del proyecto, en el próximo capítulo se desarrollará el análisis del problema y se creará su modelo de dominio.

Capítulo 4

Modelo de análisis

En este capítulo se detalla el análisis de la aplicación. Debido a que ya está diseñada e implementada no se hará un nuevo análisis, pues se partirá de la versión anterior.

4.1. Casos de uso del sistema

Los casos de uso se definen como una secuencia de acciones realizadas por el sistema, que producen un resultado valioso para un actor en particular [8].

Gracias a los casos de uso podemos describir el comportamiento del sistema y como interacciona con el usuario (actor).

Además cumplen tres objetivos fundamentales: deben satisfacer los requisitos funcionales, sirven de guía para la fase de diseño y ayudan a seleccionar las pruebas críticas, dónde se puede dar algún error.

4.1.1. Diagramas de casos de uso

La figura 4.1 corresponde al diagrama de casos de uso. En ella pueden observarse todas las interacciones entre actor y sistema.

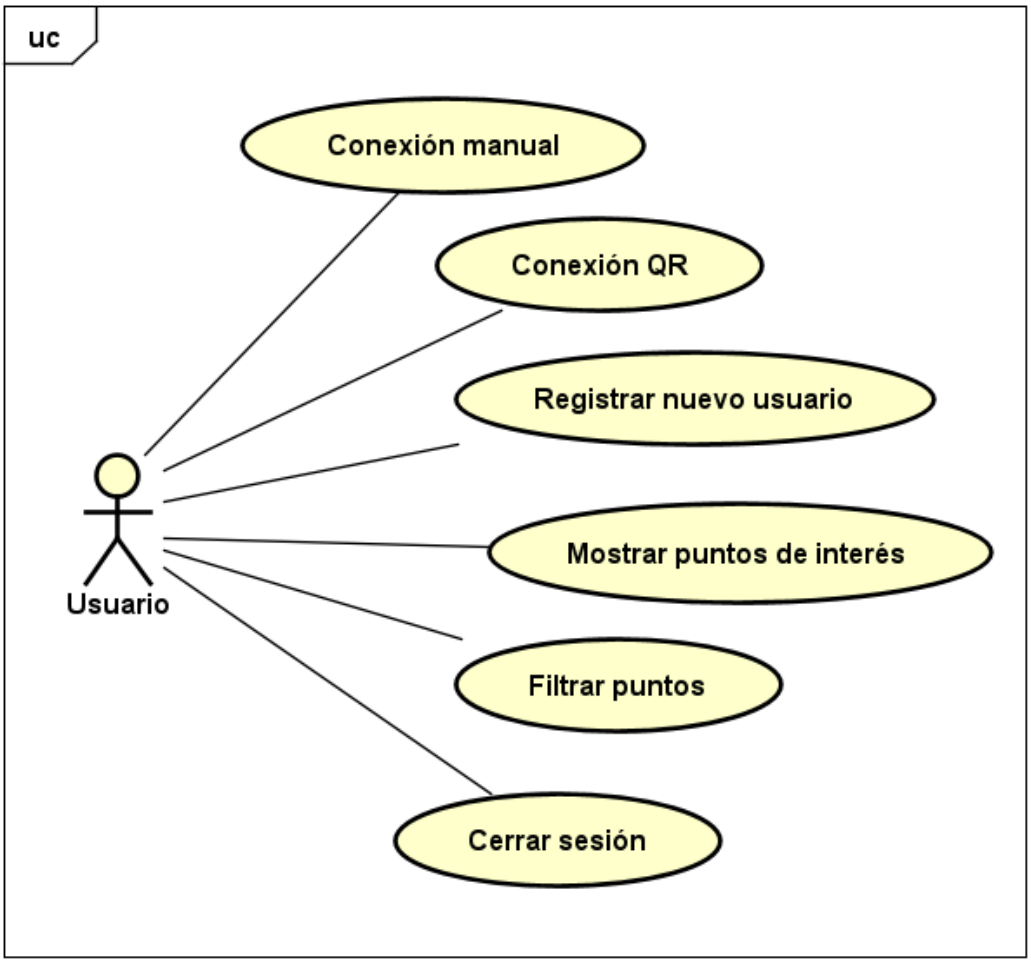


Figura 4.1: Casos de Uso de la Plataforma

A continuación se explican con detalle los casos de uso del diagrama anterior.

Caso de uso 1: Conexión manual

Mediante este caso de uso el usuario inicia sesión en la aplicación utilizando sus credenciales, usuario y contraseña. Necesario para que *Xtremeloc* conozca nuestra posición.

CU-1	Conexión Manual
Actor	Usuario
Descripción	El usuario inicia sesión para obtener la información de los elementos que puede observar.
Precondición	El usuario con el que se quiere iniciar sesión tiene que existir.
Postcondición	El sistema registra la conexión con el servidor.
Secuencia Normal	**1.** El usuario indica que quiere realizar una conexión manual. **2.** El sistema solicita al usuario la información de acceso al servidor y sus credenciales. **3.** El usuario introduce la ruta de acceso al servidor y sus credenciales y selecciona iniciar sesión. **4.** El sistema comprueba las credenciales y conecta con el servidor.
Excepciones	**3.1.** El usuario cancela. El CU termina. **4.1.** Las credenciales no son correctas. El sistema muestra un mensaje de error y el CU continúa en el paso 3.

Tabla 4.1: Caso de Uso 1 - Conexión manual

Caso de uso 2: Conexión QR

Mediante este caso de uso el usuario inicia sesión en la aplicación utilizando un código QR. Necesario para que *Xtremeloc* conozca nuestra posición.

CU-2	Conexión QR
Actor	Usuario
Descripción	El usuario inicia sesión para obtener la información de los elementos que puede observar.
Precondición	El código QR con el que se quiere iniciar sesión tiene que existir.
Postcondición	El sistema registra la conexión con el servidor.
Secuencia Normal	**1.** El usuario indica que quiere realizar una nueva conexión mediante QR. **2.** El sistema muestra la visión de la cámara trasera. **3.** El usuario centra un código QR en la visión de la cámara. **4.** El sistema escanea el código y conecta con el servidor.
Excepciones	**3.1.** El usuario cancela, el caso de uso termina. **4.1.** El código QR no contiene la información correspondiente a un servidor válido, el sistema muestra un mensaje de error y el CU continúa en el paso 2.

Tabla 4.2: Caso de Uso 2 - Conexión QR

Caso de uso 3: Cerrar sesión

Mediante este caso de uso el usuario cierra sesión en la aplicación.

CU-3	**Cerrar Sesión**
Actor	Usuario
Descripción	El usuario cierra la sesión activa.
Precondición	El usuario ha iniciado sesión.
Postcondición	Se cancela la conexión con el servidor.
Secuencia Normal	**1.** El usuario cierra su sesión. **2.** El sistema se desconecta del servidor y muestra la pantalla de inicio de sesión.
Excepciones	Ninguna.

Tabla 4.3: Caso de Uso 3 - Cerrar Sesión

Caso de uso 4: Registrar nuevo usuario

Mediante este caso de uso el actor crea un nuevo usuario con el que iniciar sesión en el servidor.

CU-4	**Registrar nuevo usuario**
Actor	Usuario
Descripción	El usuario crea un nuevo usuario con el que conectarse al servidor.
Precondición	Ninguna
Postcondición	La información sobre el nuevo usuario queda registrada en el sistema y el usuario inicia sesión.
Secuencia Normal	**1.** El usuario indica que quiere registrarse como nuevo usuario. **2.** El sistema muestra el formulario para la creación de un nuevo usuario. **3.** El usuario rellena el formulario y confirma el registro. **4.** El sistema guarda la información e inicia sesión.
Excepciones	**3.1** El usuario cancela. Se cierra el formulario y el CU queda sin efecto. **4.1** El sistema detecte algún dato incorrecto. El sistema muestra un mensaje de error y el CU continúa en el paso 3.

Tabla 4.4: Caso de Uso 4 - Registrar nuevo usuario

Caso de uso 5: Mostrar puntos de interés

Mediante este caso de uso el sistema muestra los puntos de interés en la pantalla del dispositivo móvil

CU-5	Mostrar puntos de interés
Actor	Usuario.
Descripción	El sistema muestra los puntos de interés sobre la imagen de la cámara trasera que aparece en la pantalla
Precondición	El usuario está registrado.
Postcondición	Ninguna.
Secuencia Normal	**1.** El usuario indica que quiere observar los puntos de interés. **2.** El sistema muestra la visión de la cámara trasera superponiendo los elementos registrados como observables, diferenciando entre los que entran en el campo de visón de la cámara y los que no.
Excepciones	**2.1.** El usuario cancela, el UC termina.

Tabla 4.5: Caso de Uso 5 - Mostrar puntos de interés

Caso de uso 6: Filtrar puntos

Mediante este caso de uso el selecciona los puntos que quiere que se muestren.

CU-6	Filtrar puntos
Actor	Usuario
Descripción	Seleccionar y filtrar los puntos a observar.
Precondición	El usuario está registrado.
Postcondición	Los nuevos puntos aparecen sobre la visión de la cámara trasera del dispositivo.
Secuencia Normal	**1.** El usuario indica que quiere filtrar los puntos a observar. **2.** El sistema muestra una lista con los puntos observables por el usuario. **3.** El usuario selecciona los elementos a observar y acepta. **4.** El sistema guarda los puntos a mostrar.
Excepciones	**3.1.** El usuario cancela, el CU queda sin efecto.

Tabla 4.6: Caso de Uso 6 - Filtrar puntos

4.2. Modelo de dominio

En la figura 4.2 se muestra el modelo de dominio del sistema, en en el que aparecen las clases existentes a nivel de análisis.

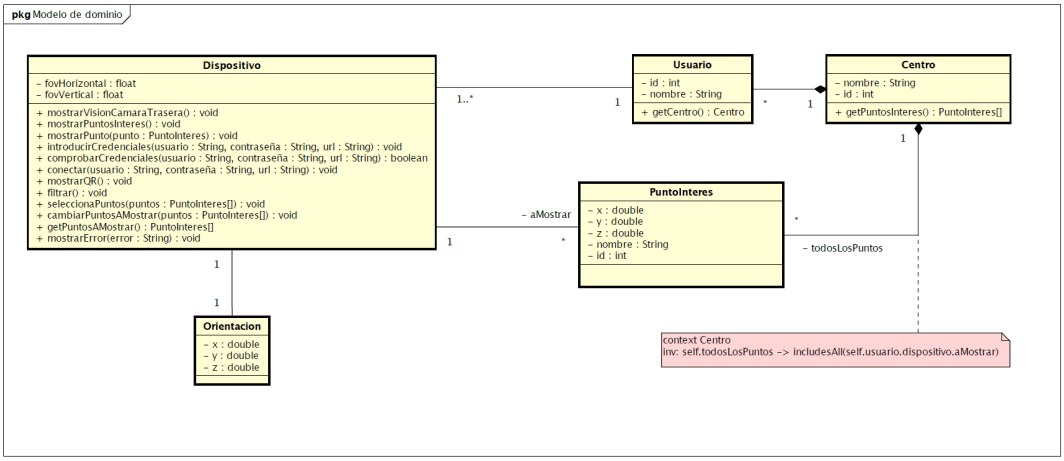


Figura 4.2: Modelo de Dominio

Después de realizar el análisis del problema, en el siguiente capítulo se describirán las tareas llevadas a cabo durante la fase de diseño.

Capítulo 5

Diseño

En este capítulo se explica el modelo de diseño de la aplicación, partiendo de su modelo de análisis, la arquitectura software del sistema, y las decisiones relativas al diseño que se han tomado para desarrollar la aplicación.

5.1. Arquitectura del Sistema

Se ha decidido mantener la misma arquitectura que la aplicación de partida. *Skywalker* se basa en una arquitectura de 3 capas:

- Modelo en la que se encuentra la lógica de la aplicación.
- Vista a ella pertenecen los *layouts*.
- Controlador en la que se encuentran las actividades.

Además de estás capas se han añadido otras dos:

- Servicios con utilidades matemáticas como vectores o matrices.
- Persistencia donde se realizan las comunicaciones con la API de *Xtremeloc* utilizando el patrón Fachada.

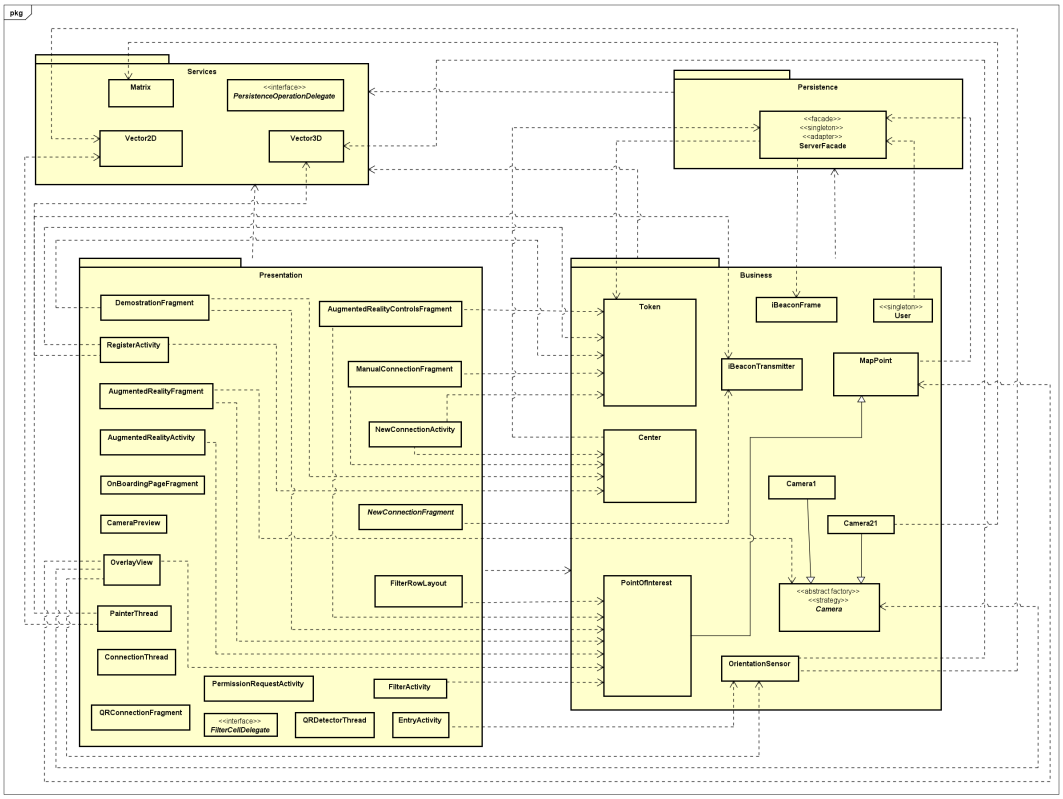


Figura 5.1: Arquitectura General del Sistema

5.2. Interfaz de usuario

Se ha diseñado la interfaz de usuario utilizando los *fragmentos Android* [9], que permiten reutilizar elementos de la interfaz y permiten que nuevas vistas reutilicen la ya creadas. También favorecen la cohesión de las clases controladoras.

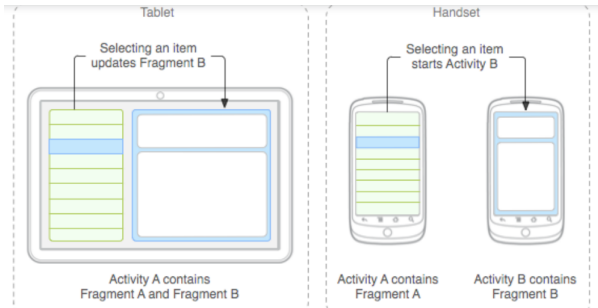


Figura 5.2: Fragmentos Android

5.2.1. Bocetos interfaz de usuario

A continuación se muestran los bocetos de la interfaz de usuario realizados para comprobar que ésta, una vez se corregían los diferentes fallos por los que no funcionaba la App, presentaba un aspecto similar y satisfacía los requisitos mencionados en el Capítulo 2. También se ha realizado el boceto de la interfaz de usuario para la nueva funcionalidad: Registro de nuevo usuario. Se han obviado aquellas interfaces que tienen que ver con instrucciones de uso y permisos.

Se han realizado mediante la herramienta online Balsamiq.

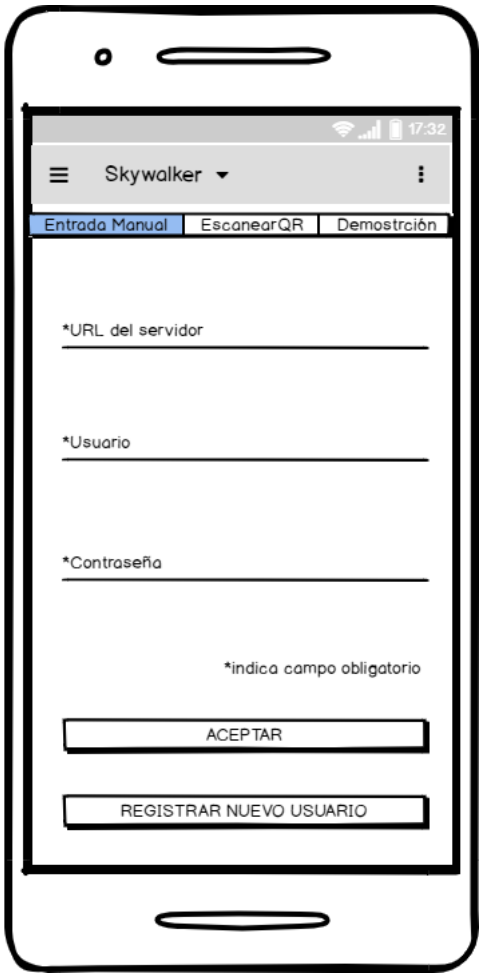


Figura 5.3: Entrada manual

Figura 5.4: Escanear código QR

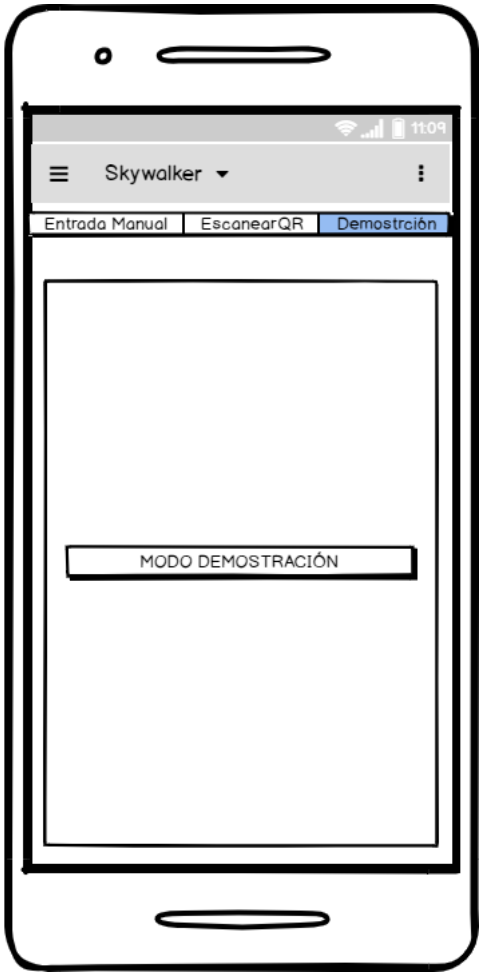


Figura 5.5: Modo demostración

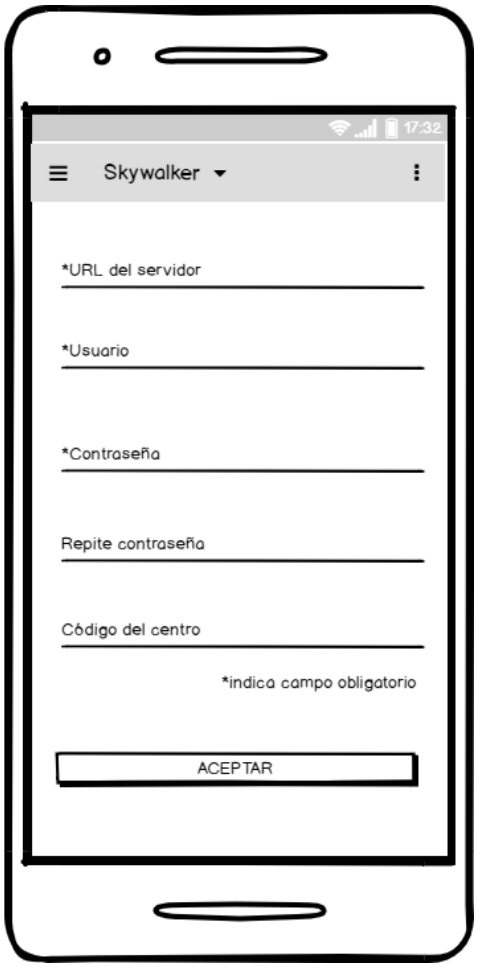


Figura 5.6: Registro de usuario

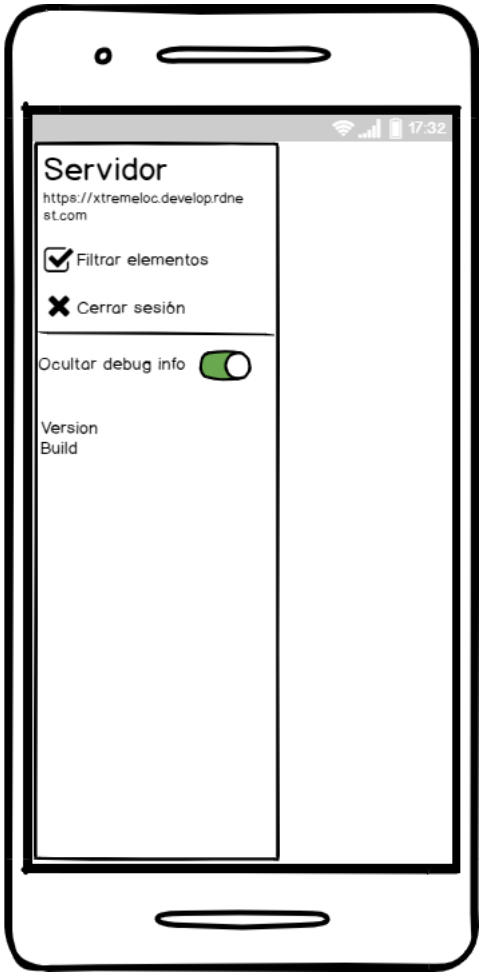


Figura 5.7: Menú lateral

Figura 5.8: Seleccionar elementos

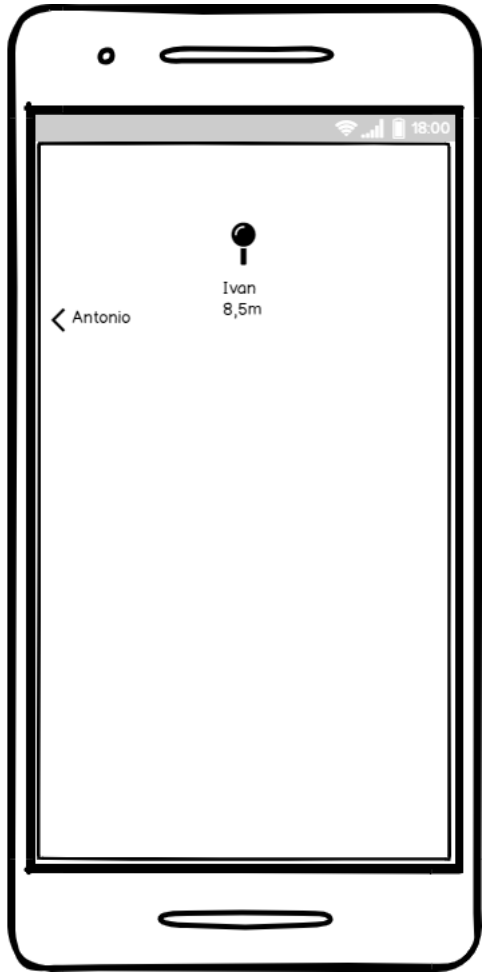


Figura 5.9: Cámara

5.3. Concurrencia

Las peticiones al servidor y los cálculos matemáticos deben ser invisibles al usuario, éste no debe percibir ningún retraso en los tiempos de respuesta de la aplicación. Esto se consigue mediante el uso de hilos de ejecución.

Gracias a estos hilos se puede repartir la carga de trabajo, la única limitación para los hilos Android es que el hilo principal es el único que puede manejar las vistas y las interacciones con el usuario [10].

Teniendo esto en cuenta, se ha decidido mantener la distribución de los hilos tal y cómo estaba en la aplicación original:

- **Hilo principal:** encargado del dibujado de la interfaz y de la interacción con el usuario.

- **Hilo de la cámara:** encargado de mostrar la visión de la cámara trasera en pantalla.

- **Hilo pintor:** encargado de dibujar los elementos en pantalla.
- **Hilo del sensor de orientación:** encargado de gestionar los cambios en el sensor de orientación, realizando los cálculos necesarios para conocer la dirección actual del dispositivo.
- **Hilo *Bluetooth LE***: encargado de enviar la señal *Bluetooth* a las antenas.
- **Hilo emisor**: encargado de ejecutar las peticiones al servidor y actualizar los datos.

Android admite tanto la utilización de *framworks* y métodos especiales como el uso de los tradicionales hilos y semáforos para la generación de peticiones asíncronas que permitan el reparto del trabajo entre el hilo principal y los secundarios.

Se ha decidido mantener los hilos que realizan el dibujado y la emisión de peticiones al servidor como hilos de bajo nivel, que aunque no estén pensados para un uso continuado poseen planificadores que se encargan del uso eficiente de los hilos; y para el resto se utilizarán los *frameworks* de paralelismo proporcionados por la plataforma por su facilidad de uso y porque la mayoría de constructores de otros *frameworks* utilizados toman como parámetro algún tipo de hilo o cola de operaciones.

5.4. Sensor de orientación

Como se explica en el TFG de partida, el objetivo es conseguir un vector normal al dispositivo, donde X apunte al norte magnético, Z apunte al suelo, siendo Y ortogonal a ambos y apuntando al este magnético, todo ello representado en coordenadas del mapa proporcionado por *Xtremeloc*.

5.4.1. Matrices de rotación

Los dispositivo Android tienen un tipo especial de sensor que proporciona la orientación espacial en matrices de rotación, entre otras representaciones.

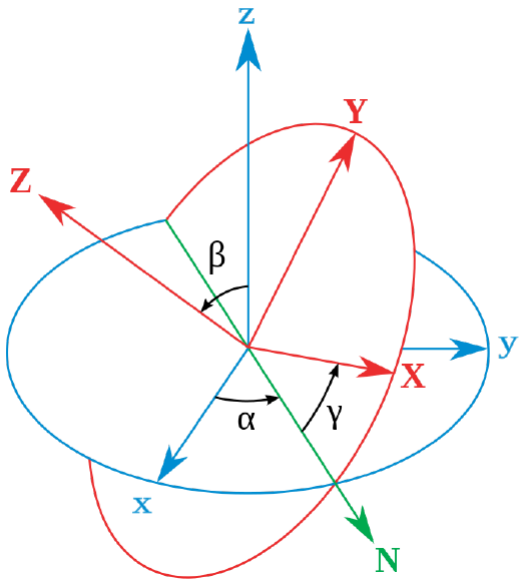


Figura 5.10: Sensor de orientación en dispositivos

Tomando como referencia la figura 5.10 se obtiene la siguiente matriz de rotación:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Se ha elegido esta representación debido a que es la más fácil de programar, y aunque haya otras que ocupan menos espacio en memoria y computacionalmente más eficientes, al haber una única representación activa en cada instante, se puede trabajar con este inconveniente. Además, estas matrices presentan la siguiente propiedad:

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma)$$

Siendo R una rotación cuyos ángulos tait-bryan son α, β y γ sobre los ejes z, y, x respectivamente [11]. Para rotar una matriz de rotación sobre un eje arbitrario, simplemente si tiene que realizar el siguiente producto:

$$R' = RR_e$$

Siendo R_e la matriz de rotación sobre un eje arbitrario.

5.4.2. Vector normal al dispositivo

El marco de referencia de los dispositivos no está preparado para trabajar con el dispositivo en vertical, sino en horizontal, como si estuviera apoyado sobre una superficie.

Para rotar la matriz de coordenadas se utiliza el vector $V_r = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$, ya que el *framework* tiene como referencia un estado horizontal.

Teniendo en cuenta el funcionamiento de las matrices de rotación, se deduce que llevar un punto del estado original al nuevo estado descrito por dicha matriz no necesita realmente de ninguna transformación para la matriz, si somos capaces de obtener el correspondiente punto a transformar descrito en términos de la referencia utilizada por la matriz.

Por otro lado, para normalizar la matriz de rotación se debe multiplicar la matriz de rotación por el vector $V_n = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$, ya que Android considera que es el eje Y del marco de referencia del sensor el que apunta al norte magnético. Esta operación es equivalente a extraer la segunda columna de la matriz y computacionalmente mucho menos costosa.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} b \\ e \\ h \end{pmatrix}$$

Para obtener el vector representado en coordenadas de *Xtremeloc* bastará con aplicar al vector de orientación obtenido, el desfase existente entre el norte magnético real y norte magnético representado en coordenadas de *Xtremeloc*, el cuál se obtiene mediante la API REST del sistema.

5.5. Dibujado de elementos en pantalla

5.5.1. Problemática

Para dibujar los elementos en pantalla se necesita conocer:

- El vector de orientación del dispositivo, adaptado a las coordenadas utilizadas por *Xtremeloc*.
- El campo de visión de la cámara.
- La posición de cada elemento a dibujar.
- La posición del dispositivo.

El vector de orientación ya está disponible gracias al sensor de orientación, al igual que el ángulo de visión de la cámara. Las posiciones, tanto del dispositivo como de los demás elementos, también son conocidas por la API de *Xtremeloc*. Sólo falta componer la información para mostrarla en pantalla.

Por cada elemento a mostrar es necesario realizar un vector desde las coordenadas del dispositivo hacia las coordenadas del punto. Gracias al vector de orientación, se obtiene el ángulo entre la orientación del usuario y el punto a mostrar.

Se debe comprobar si dicho ángulo es menor al ángulo de visión horizontal de la cámara, si es así, el punto entra dentro de la visión de la cámara, y por tanto debe ser dibujado como elemento en el campo de visión, en caso contrario, se dibuja como elemento fuera del campo de visión.

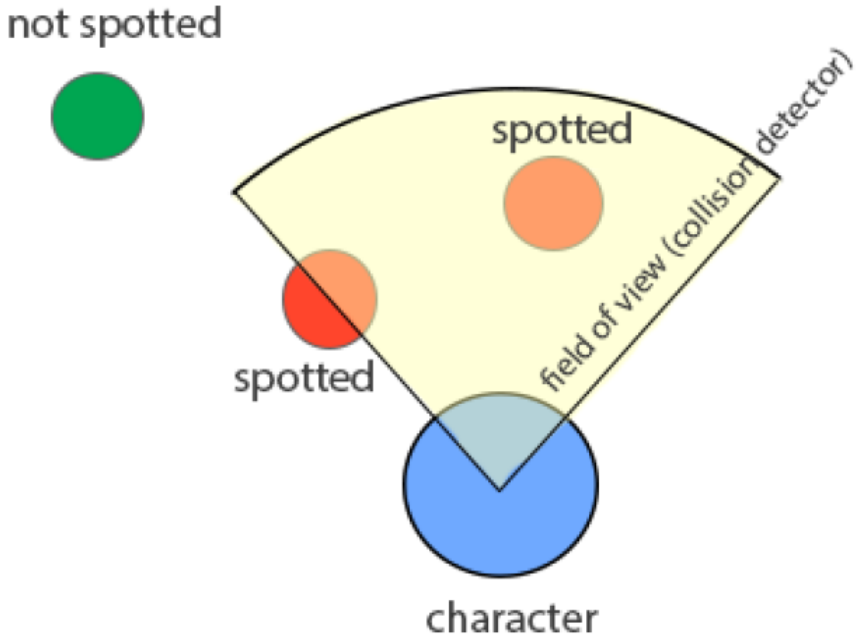


Figura 5.11: Ángulo de visión de la cámara

5.5.2. Solución para dispositivos

Para conseguir el campo de visión de la cámara trasera de un dispositivo Android, se ha decidido mantener la solución utilizada en el TFG del que se parte: utilizar la API de la cámara en conjunto con trigonometría.

Es una solución universal y funciona sin interacción del usuario, como principal inconveniente confía en que los datos del sensor de la cámara sean correctos.

Se han mantenido las dos APIs distintas de la cámara, dada la necesidad de soportar versiones antiguas, la obtención del campo de visión tendrá dos variantes:

- En la versión 1 de la API, se tienen llamadas que directamente dan los valores del campo de visión, tanto vertical como horizontal, por lo que en esta API no necesitamos trigonometría.
- Sin embargo, la versión 2 de la API prescinde de estos métodos, a cambio de otorgar las características del sensor, gracias a las cuáles, realizando unos sencillos cálculos se podrán obtener los ángulos deseados.

$$\alpha = 2 \arctan \frac{L}{2f_c}$$

donde L es la dimensión del objetivo, y f_c la longitud focal de la lente [12].

La información del sensor dada por la API nos proporciona estos datos, divididos en categoría horizontal y vertical. Dado que no se va a permitir hacer zoom sobre la imagen, y el foco va a estar gestionado por la API no se necesitan cálculos adicionales.

5.6. Transmisión de señal *Bluetooth*

Como se explica en la Sección 1.2, las antenas de *Xtremeloc* se comunican con dispositivos *Bluethooth* para conocer su posición.

Android permite el envío de este tipo de señales, por lo que se tiene que crear un objeto compatible con el estándar *iBeacon*.

5.7. Peticiones al servidor

El servidor de *XtremeLoc* responde a peticiones tipo REST enviadas desde *Skywalker*,éstas se realizan desde un hilo extra. Toda la información sobre estas operaciones se encuentra en la Sección 6.2.

5.8. Modelo de dominio en diseño

A continuación se muestra el diagrama UML correspondiente al modelo de dominio en la fase de diseño.

Se ha mantenido la división del modelo de dominio en varias imágenes para su mejor comprensión:

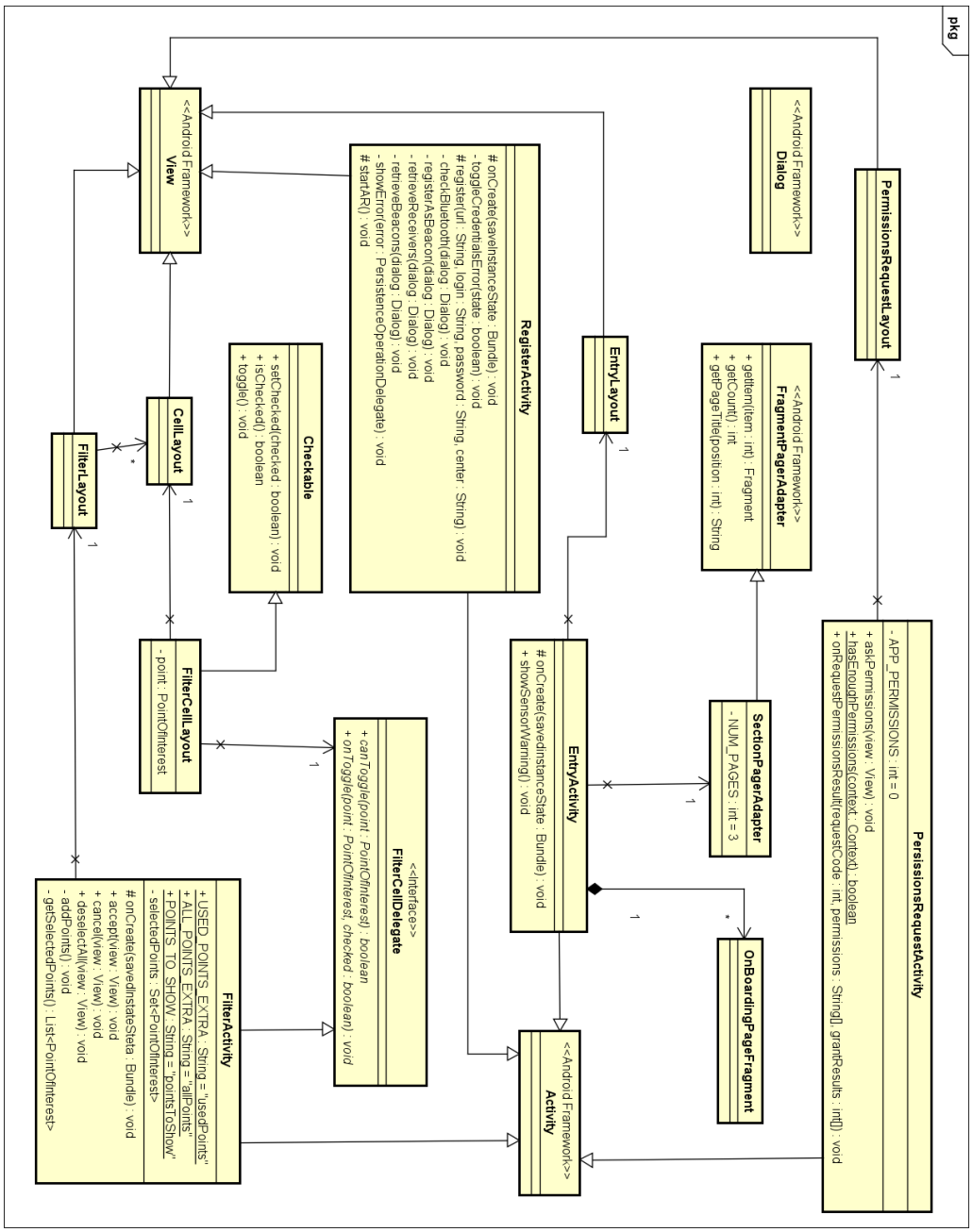


Figura 5.12: Diagrama de clases para la capa de presentación I

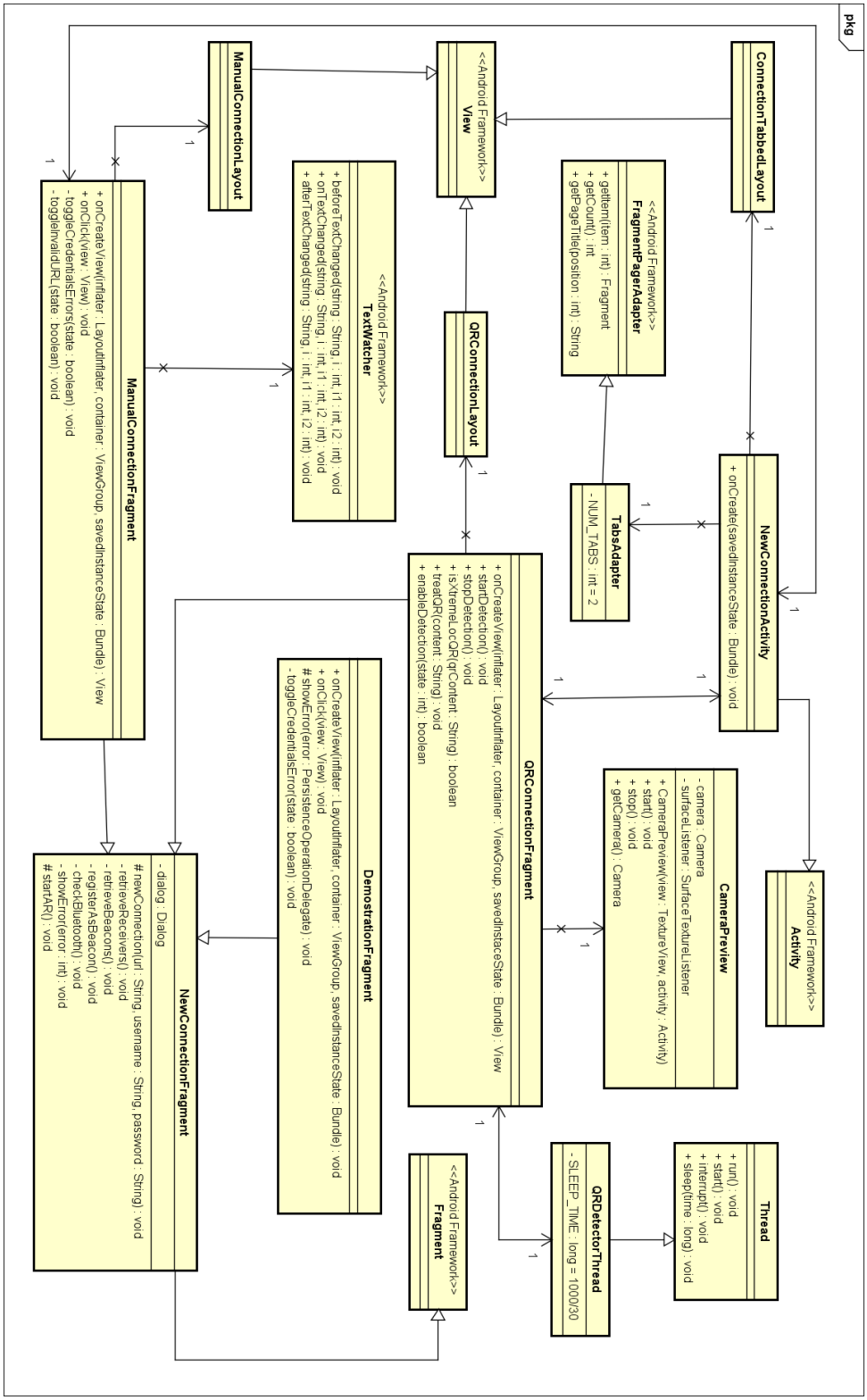


Figura 5.13: Diagrama de clases para la capa de presentación II

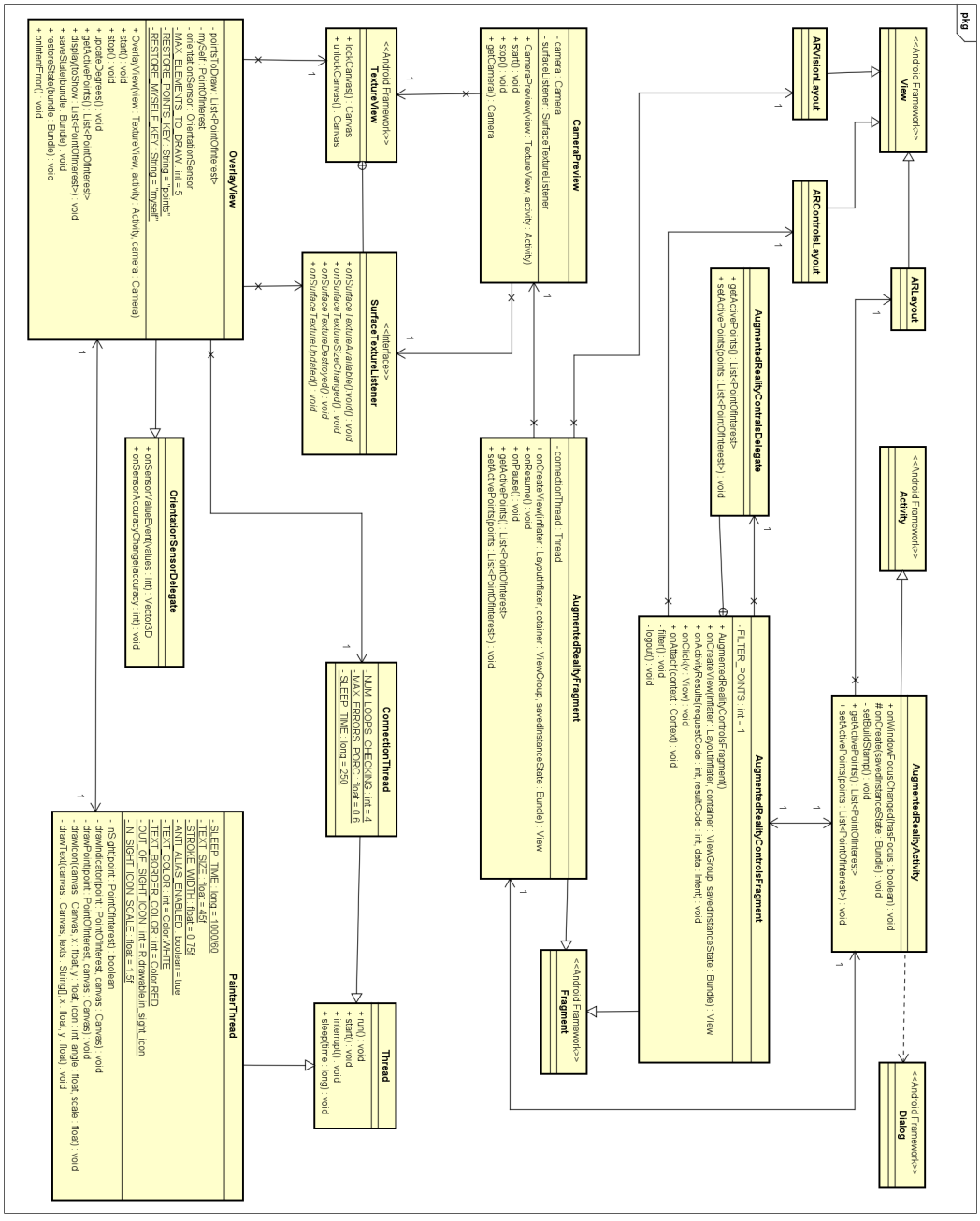


Figura 5.14: Diagrama de clases para la capa de presentación III

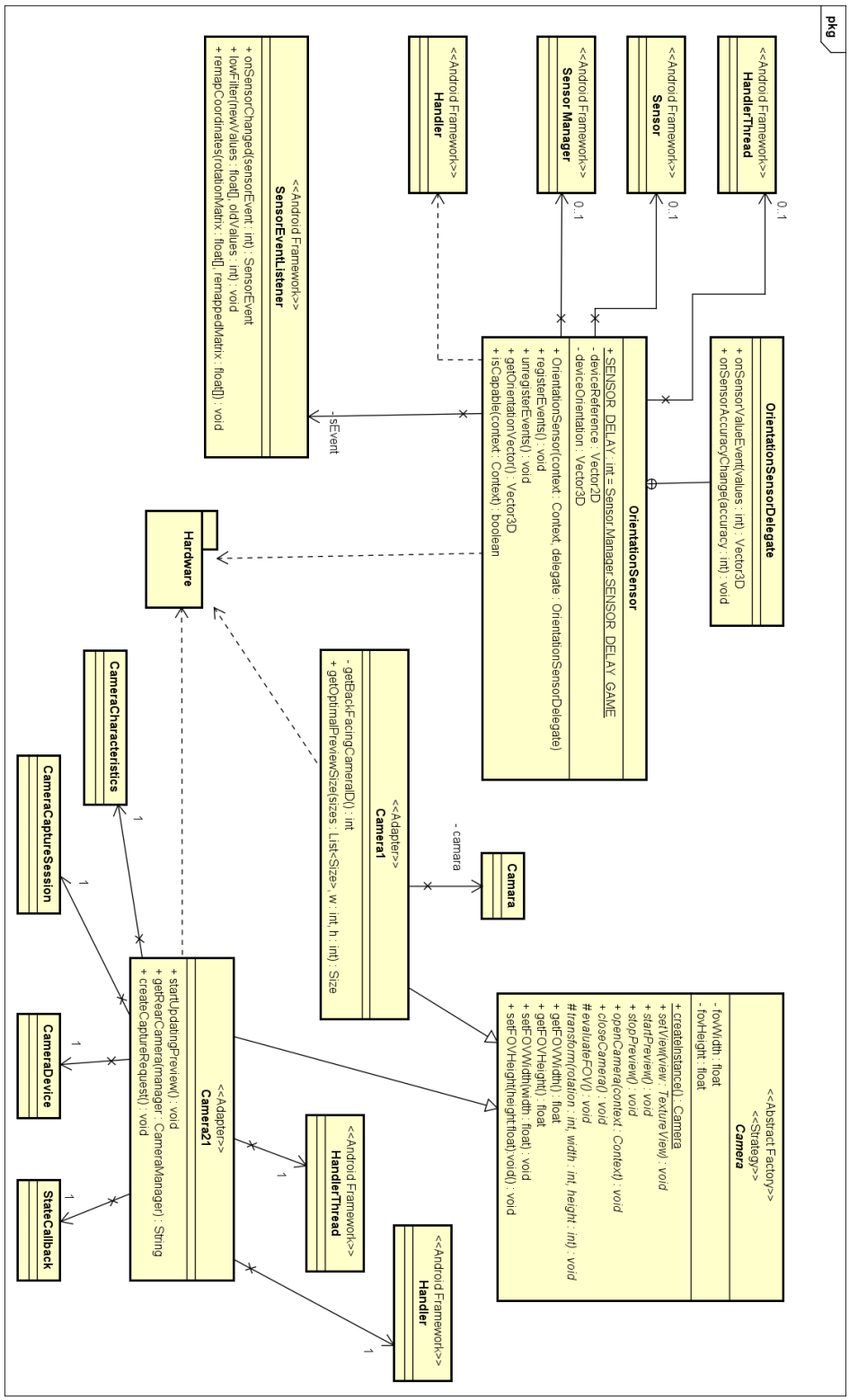


Figura 5.15: Diagrama de clases para la capa de negocio I

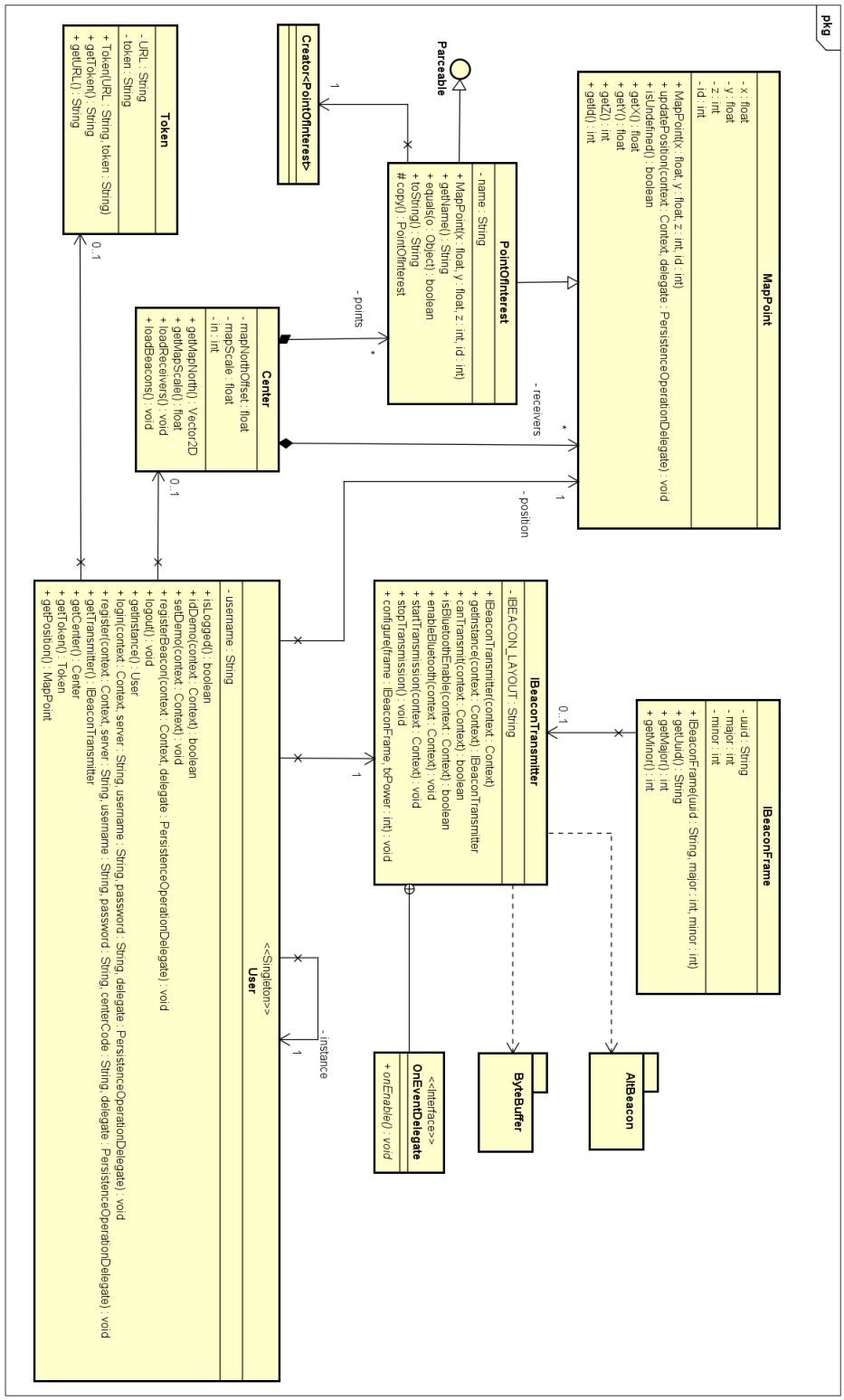


Figura 5.16: Diagrama de clases para la capa de negocio II

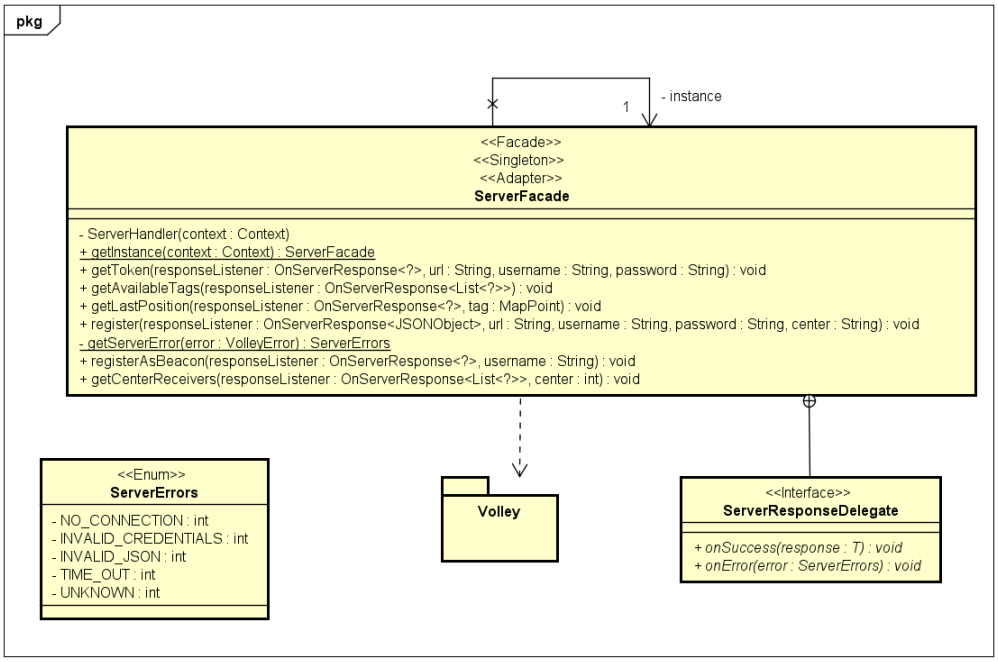


Figura 5.17: Diagrama de clases para la capa de persistencia

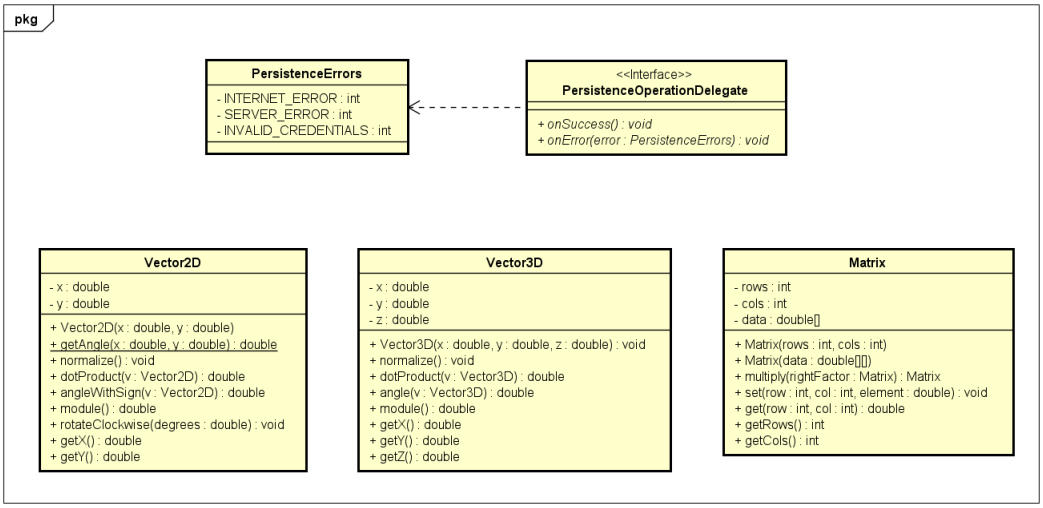


Figura 5.18: Diagrama de clases para la capa de servicios

5.9. Diagramas de secuencia en diseño

En esta sección se muestran los diagramas de secuencia, siguiendo el *standar* UML, para los casos de uso que se han considerado más relevantes para comprender el funcionamiento de la aplicación. Estos diagramas permiten ver las interacciones entre las clases de las diferentes capas.

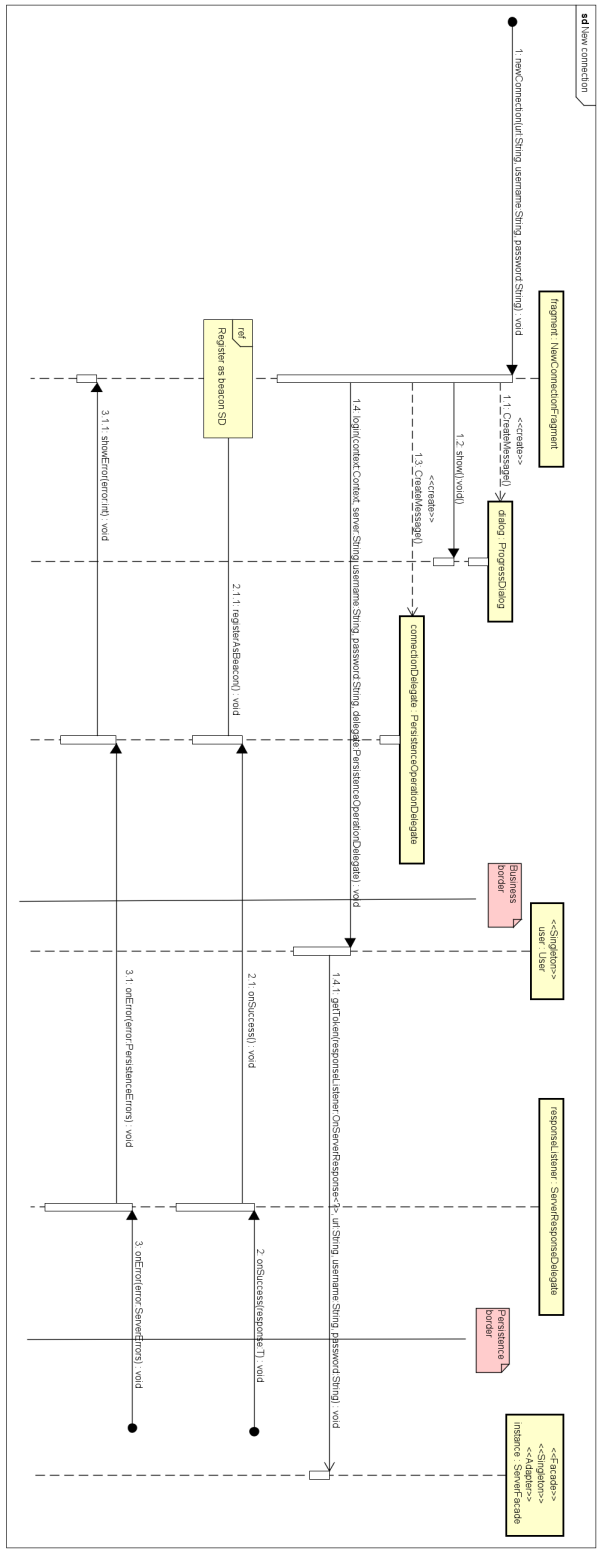


Figura 5.19: Diagrama de secuencia para una nueva conexión con el servidor con un usuario existente.

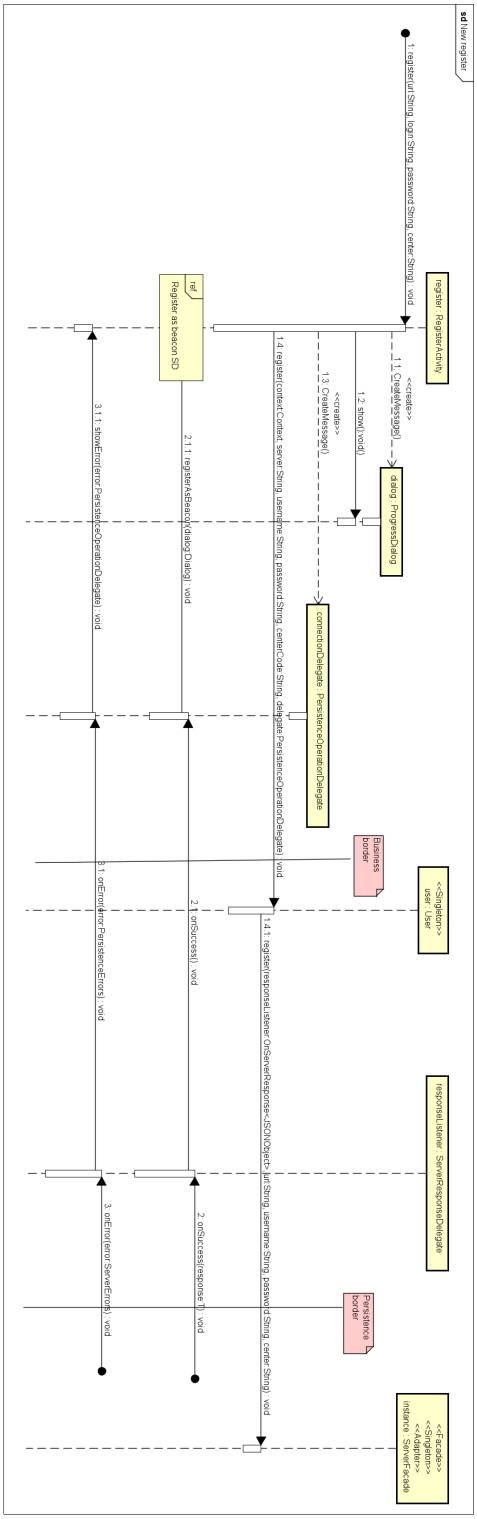


Figura 5.20: Diagrama de secuencia para una nueva conexión con el servidor con un nuevo usuario.

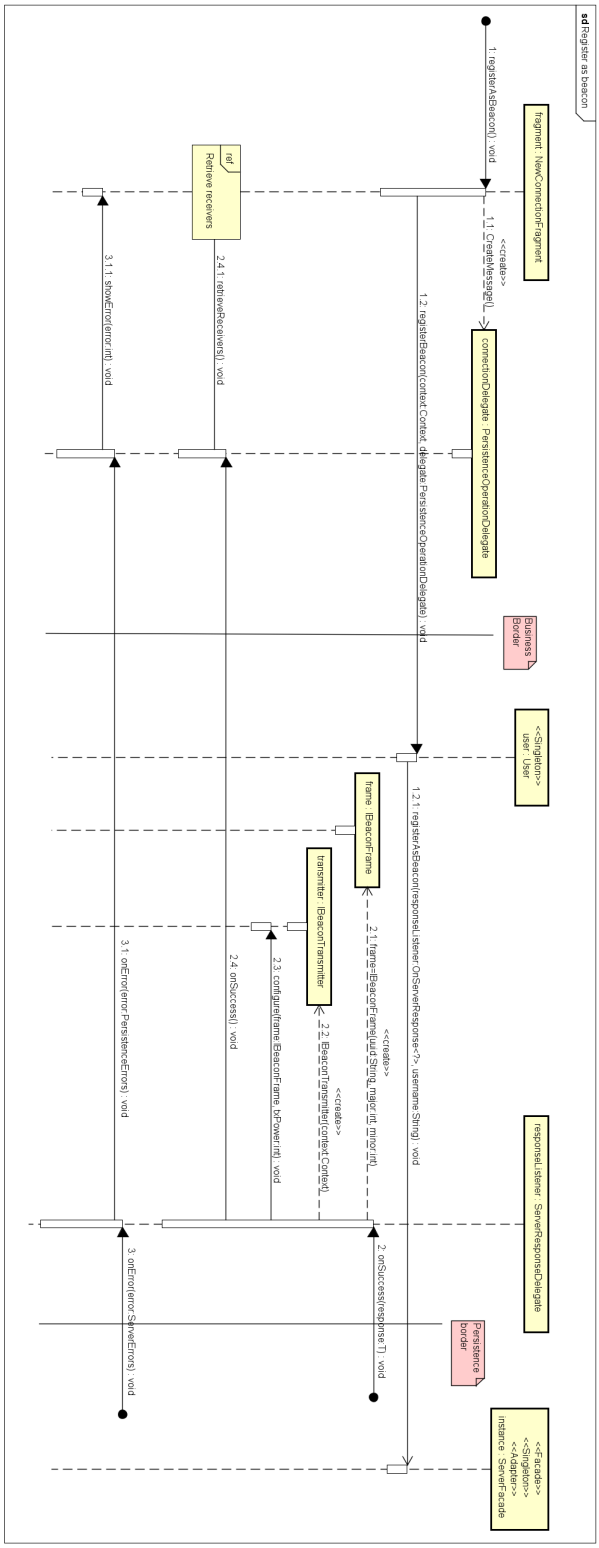


Figura 5.21: Diagrama de secuencia para el registro del dispositivo en el servidor.

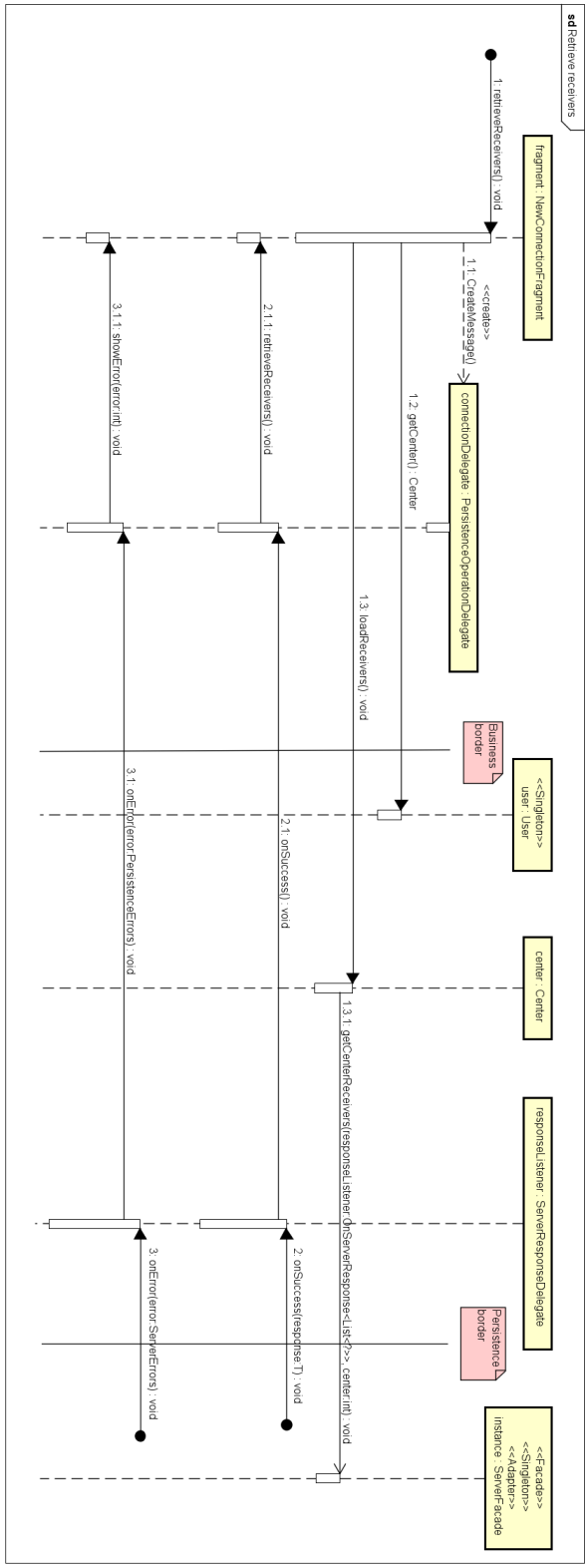


Figura 5.22: Diagrama de secuencia para la obtención de las antenas.

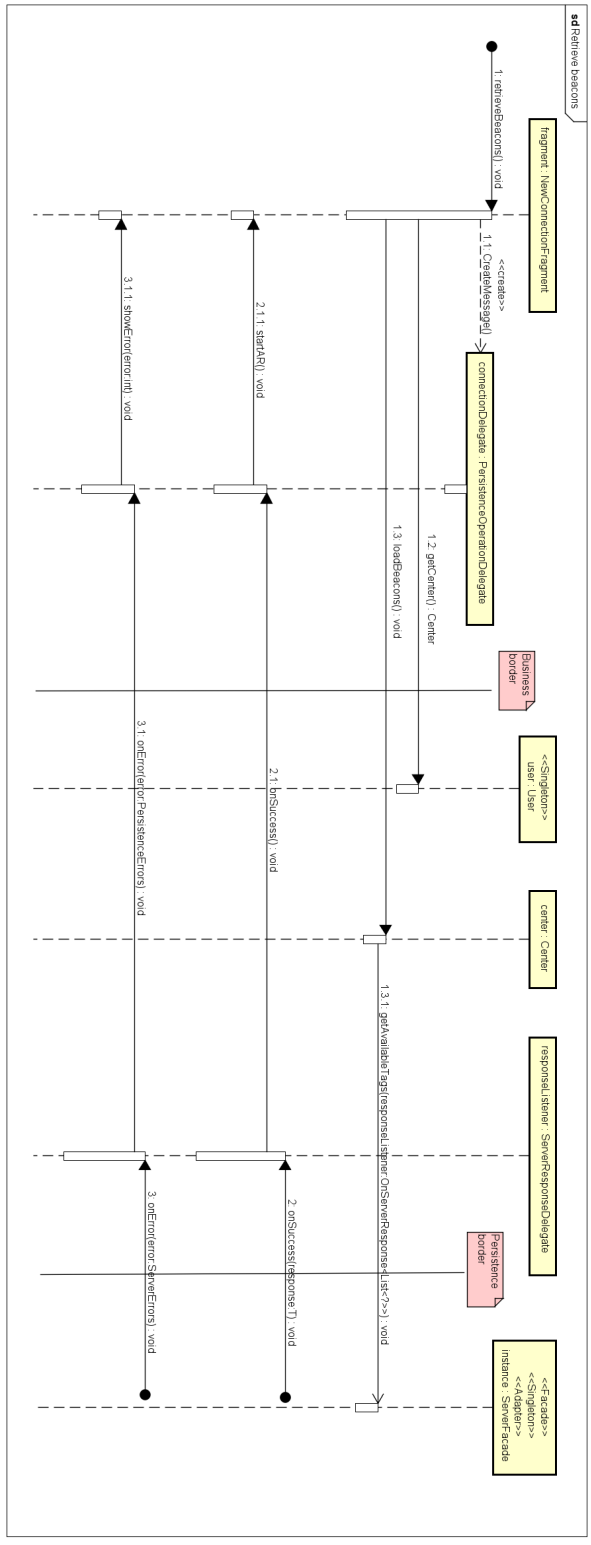


Figura 5.23: Diagrama de secuencia para la obtención de los *beacons*.

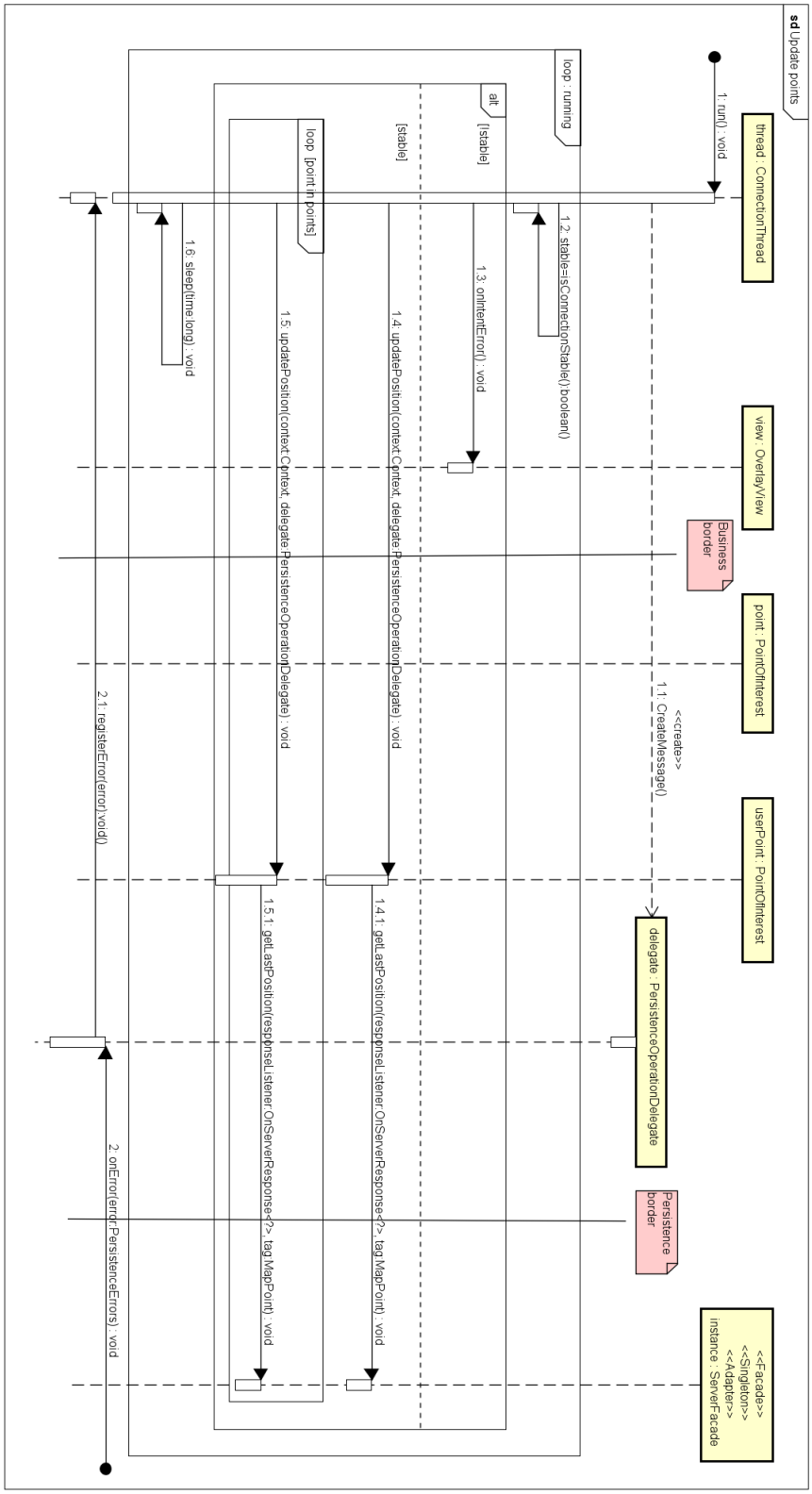


Figura 5.24: Diagrama de secuencia para actualizar la posición de un *beacon*.

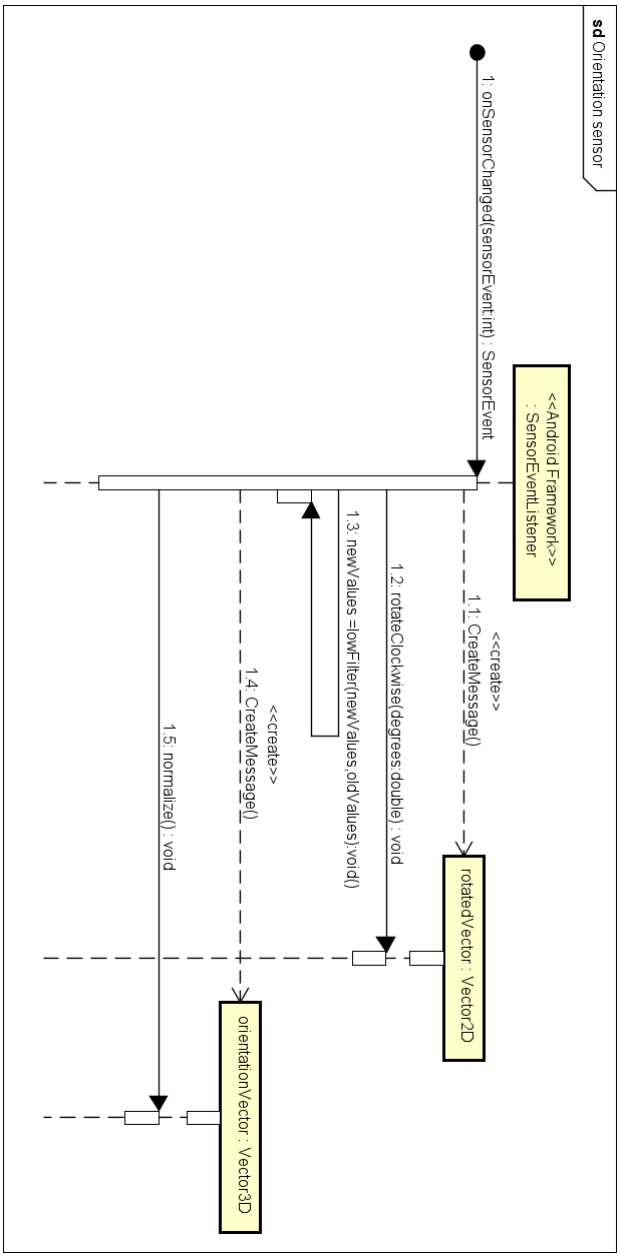


Figura 5.25: Diagrama de secuencia correspondiente al hilo que gestiona los cambios en la orientación del dispositivo.

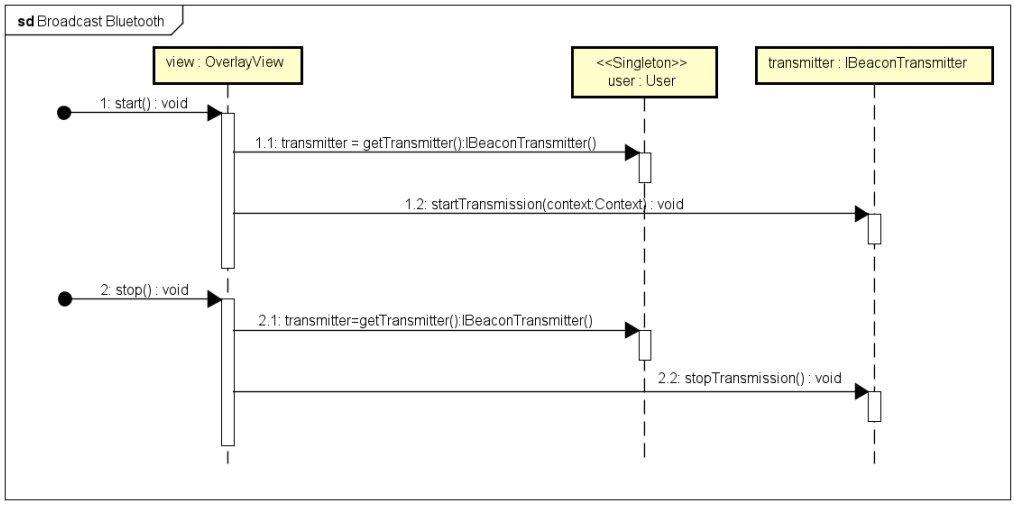


Figura 5.26: Diagrama de secuencia para la gestión de la señal *Bluetooth* a emitir.

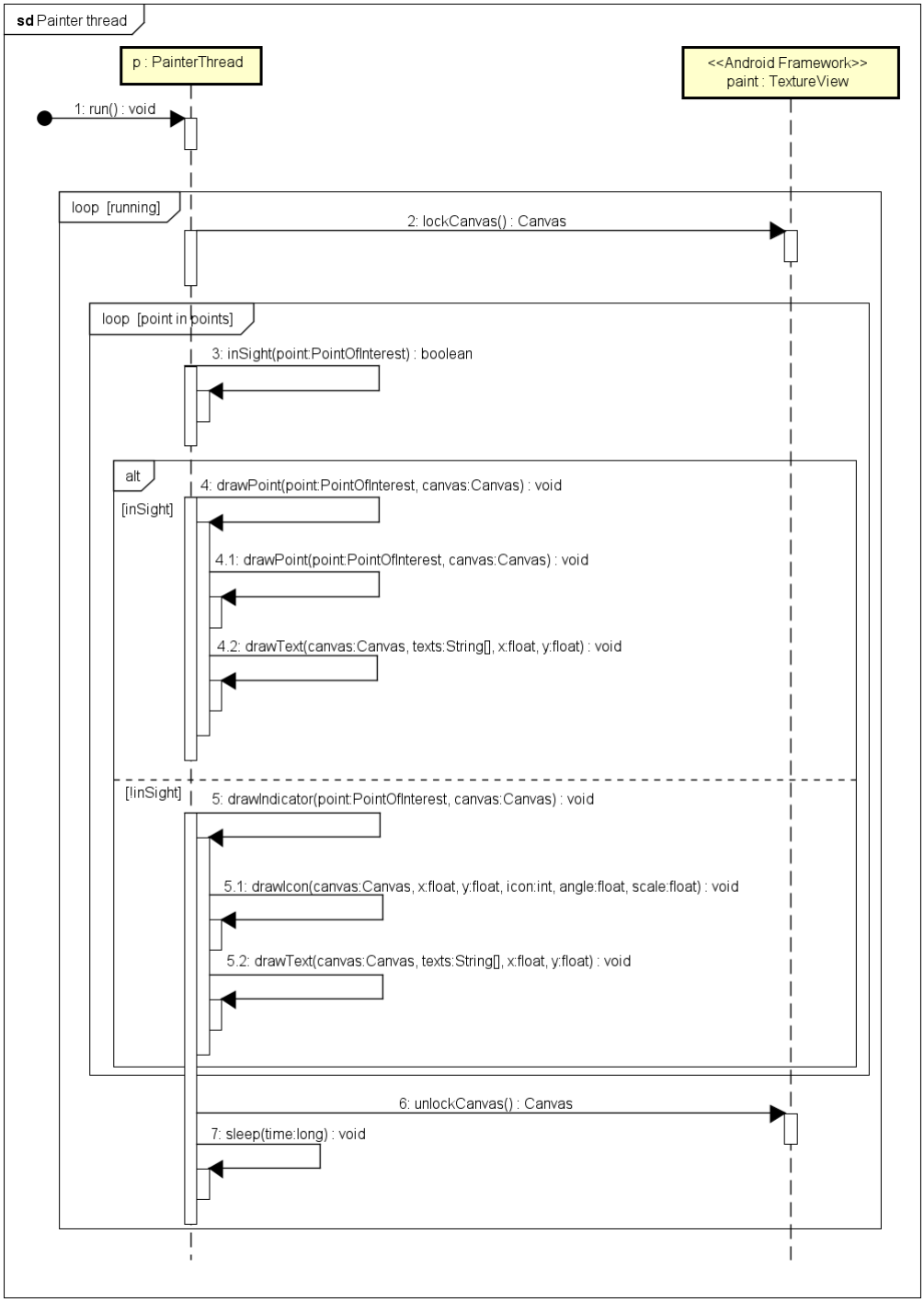


Figura 5.27: Diagrama de secuencia correspondiente al hilo que gestiona el dibujado en pantalla.

5.10. Diagrama de operaciones REST

En esta sección aparece el diagrama de las operaciones de la API REST de XtremeLoc que utiliza la aplicación.

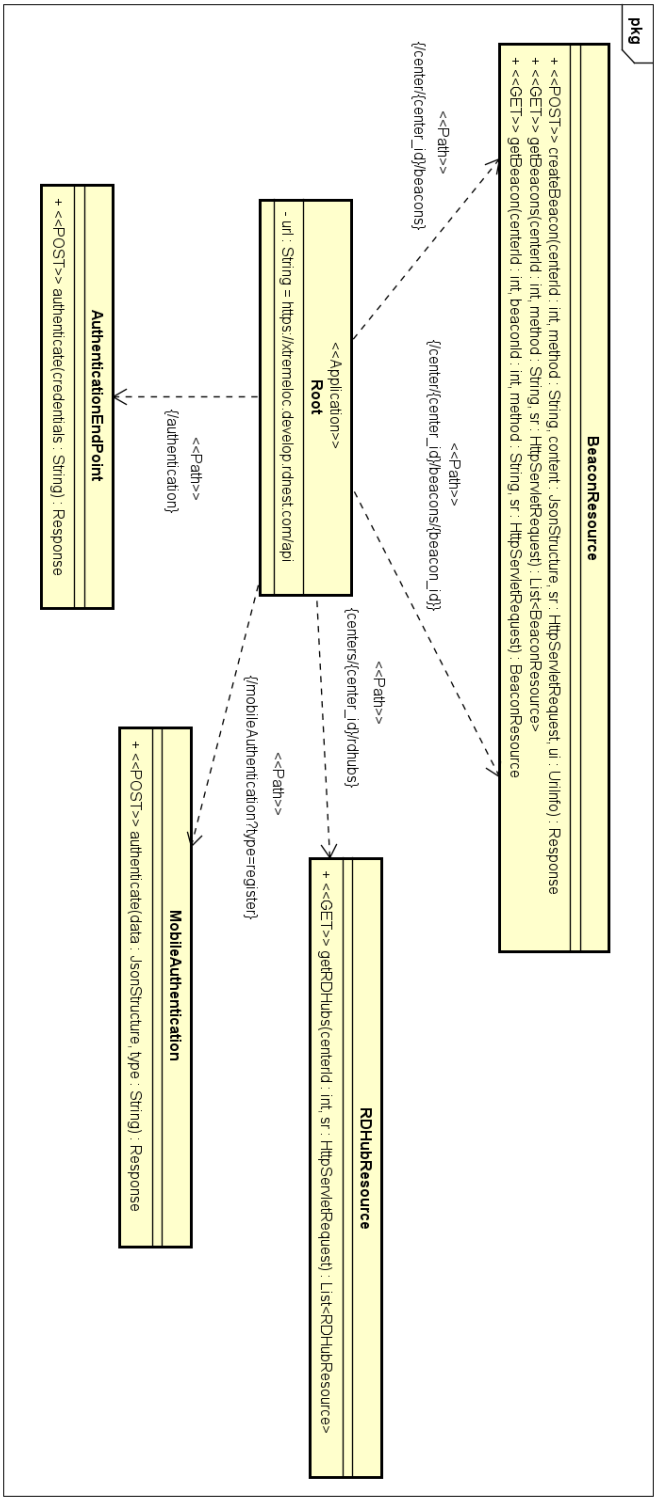


Figura 5.28: Diagrama de operaciones REST

Tras explicar el diseño de la aplicación, en el siguiente capítulo se explicarán los cambios realizados respecto a la aplicación base.

Capítulo 6

Reingeniería del sistema anterior

En este apartado se describirá todo el proceso de reingeniería de la aplicación de la que se partió.

6.1. Estado inicial del sistema

Se disponía del código fuente, que se podía compilar, pero no funcionaba correctamente. Gracias a la memoria del TFG en el que se desarrolló *Skywalker*, y al modo demostración de la aplicación, se comprendió su funcionalidad.

6.2. Operaciones REST

Todas las operaciones se realizan desde la clase `ServerFacade.java`, por tanto todas las funciones de las que se habla en esta sección pertenecen a dicha clase.

Como se ha expuesto anteriormente, *Xtremeloc* ha sufrido modificaciones desde que se realizó la aplicación y no se tuvo en cuenta que podían afectar a la App.

6.2.1. Registro del disposivo como beacon

El *end-point* que se utiliza para activar el móvil como *beacon* en *Xtremloc* cambió, por lo que la URL que se define en la función `registerAsBeacon()` ya no será:

```
String url = User.getInstance().getToken().getURL().concat(
    "/api/centers/" + User.getInstance().getCenter().getId() +
    "/tags/");
```

La nueva URL es:

```
String url = User.getInstance().getToken().getURL().concat(
    "/api/centers/" + User.getInstance().getCenter().getId() +
    "/beacons/");
```

6.2.2. Respuesta con las antenas del centro

La respuesta que devuelve *Xtremeloc* cuando se solicitan las antenas del centro, en la función `getCenterReceivers()`, ya no tiene el campo `id` sino `rdhub_id`, como puede observarse en los siguientes fragmentos de código:

```
public void onResponse(JSONArray response) {
    ...
    JSONObject json = response.getJSONObject(i);
    final int id = json.getInt("id");
    ...
}
```

```
public void onResponse(JSONArray response) {
    ...
    JSONObject json = response.getJSONObject(i);
    final int id = json.getInt("rdhub_id");
    ...
}
```

6.2.3. Solicitud de los *beacons*

El *end-point* que se utiliza para obtener los *beacons* disponibles en *Xtremeloc* cambió, por lo que la URL que se define en en la función `getAvailableBeacons()` ya no será:

```
String url = User.getInstance().getToken().getURL().concat(
    "/api/centers/" + User.getInstance().getCenter().getId()
    + "/tags");
```

La nueva URL es:

```
String url =  User.getInstance().getToken().getURL().concat(
    "/api/centers/" + User.getInstance().getCenter().getId()
    + "/beacons");
```

Además, se ha modificado el campo `id` por `beacon_id` de la respuesta recibida, por lo que dicho cambio se ha tenido que reflejar en *Skywalker*.

Por último, en la respuesta aparecían todos los *beacons* de la explotación a la que pertenece el centro, por lo que se ha añadido una comprobación (que el campo `rdhubs` no se nulo) para tener en cuenta sólo aquellos *beacons* que pertenecen al centro en el que se encuentra el dispositivo.

A continuación pueden observarse estos cambios entre el código original y el modificado:

```
public void onResponse(JSONArray response) {
    ...
    final int id = json.getInt("id");
    final String name;
    ...
    PointOfInterest point
        = new PointOfInterest(id, name);
    points.add(point);
    }
    ...
}

public void onResponse(JSONArray response) {
    ...
    final int id = json.getInt("beacon_id");
    final String name;
    final JSONArray rdhubs = json.getJSONArray("rdhubs");
    ....
    if(rdhubs.length()!=0) {
        PointOfInterest point
            = new PointOfInterest(id, name);
        points.add(point);
    }
    ...
}
```

6.2.4. Respuesta con las coordenadas de un *beacon*

El *end-point* que se utiliza para obtener las coordenadas de los beacons se modificó, por lo que la URL que se define en la función `getLastPosition()` ya no será:

```
String url = User.getInstance().getToken().getURL().concat(
"/api/centers/" + User.getInstance().getCenter().getId()
+ "/tags/" + point.getId());
```

La nueva URL es:

```
String url = User.getInstance().getToken().getURL().concat(
"/api/centers/" + User.getInstance().getCenter().getId()
+ "/beacons/" + point.getId());
```

Por otro lado, la respuesta que se obtiene ya no tiene el campo `nearest_rdhub`, se utilizará el campo `rdhubs`, como ya el *beacon* pertenece al centro, nunca el campo `rdhubs` será nulo, por eso no es necesario realizar esa comprobación.

Estas modificaciones pueden observarse en los siguientes fragmentos de código, el primero pertenece al código original y el segundo al modificado:

```
public void onResponse(JSONObject response) {
    ...
    if (!response.has("nearest_rdhub")) {
        return;
    }
    final int receiverId =
        response.getInt("nearest_rdhub");
    ...
}
```

```
public void onResponse(JSONObject response) {
    ...
    final JSONArray rdhubs = response.getJSONArray("rdhubs");
    ....
}
```

6.3. Permisos

El sistema de control de permisos que tenía *Xtremeloc* impedía conocer a un usuario móvil, rol que se da a los usuarios que sólo utilizan la App, los *beacons* del centro en el que se encontraba para así evitar problemas asociados a la privacidad de los usuarios y la seguridad del sistema.

Cómo la finalidad de la aplicación es conocer la ubicación de otras personas o elementos, se tuvo que eliminar esta restricción en *Xtremeloc*.

6.4. Construcción de la trama *iBeacon* en el dispositivo

La aplicación original utilizaba la librería `AltBeacon.beacon` para construir la trama *iBeacon* y enviarla para que fuese detectada por las antenas de *Xtremloc*.

Cómo el resto de *beacons* con los que se estaba trabajando sí aparecían en el *front-end* de *Xtremeloc* el problema era que no se envíaba correctamente la trama *iBecon* desde el dispositivo.

Se decidió utilizar un `ByteBuffer` [13] y un *Callback*, como en el resto de peticiones, aunque esta fuese una comunicación *Bluetooth* con las antenas de *Xtremeloc*, en lugar de una petición web, como las que se usan para las comunicaciones con el servidor de dicho sistema de localización.

En el siguiente fragmento de código de la clase `iBeaconTransmitter.java` puede verse cómo quedaría la función:

```
public void startTransmission(Context context) {
    BluetoothManager bluetoothManager = (BluetoothManager)context
        .getSystemService(context.BLUETOOTH_SERVICE);
    BluetoothAdapter bluetoothAdapter = bluetoothManager
        .getAdapter();
    BluetoothLeAdvertiser mBleAdvertiser = bluetoothAdapter
        .getBluetoothLeAdvertiser();

    AdvertiseSettings settings = new
        AdvertiseSettings.Builder()
            .setAdvertiseMode(AdvertiseSettings
                .ADVERTISE_MODE_BALANCED)
                    .setTxPowerLevel(AdvertiseSettings
                        .ADVERTISE_TX_POWER_MEDIUM)
                            .setConnectable(false)
                                .build();

    BatteryManager bm = (BatteryManager)context
        .getSystemService(context.BATTERY_SERVICE);

    int battery = bm.getIntProperty(
        BatteryManager.BATTERY_PROPERTY_CAPACITY);

    ByteBuffer bb = ByteBuffer.allocate(27);
    bb.put(new byte[]{(byte)0xff, 0x4c, 0x00, 0x02, 0x15});
    bb.put(hexToBytes(frame.getUUID()));
    bb.putShort((short)frame.getMajor());
    bb.putShort((short)frame.getMinor());
    bb.put((byte)-66);
    bb.put((byte)battery);

    AdvertiseData data = new AdvertiseData.Builder()
            .setIncludeDeviceName(false)
                .setIncludeTxPowerLevel(false)
                    .addManufacturerData(224, bb.array())
                        .build();

    AdvertiseCallback mCallback = new AdvertiseCallback() {
        @Override
        public void onStartSuccess(
            AdvertiseSettings settingsInEffect) {
                super.onStartSuccess(settingsInEffect);
        }
```

```
            @Override
            public void onStartFailure(int errorCode) {
                super.onStartFailure(errorCode);
            }
        };

        mBleAdvertiser.startAdvertising(settings, data, mCallback);

    }
```

6.5. Datos utilizados para el cálculo de las coordenadas

La aplicación de la que se partía asociaba a cada *beacon* las coordenadas de la antena más próxima, por lo que según se iba aproximando el dispositivo al *beacon* objetivo, las coordenadas de ambos (*beacon* asociado al dispositivo y *beacon* objetivo) coincidían y la aplicación consideraba que ya se había alcanzado el objetivo.

Para que esto no sucediese, se tuvo que considerar las coordenadas reales para hacer los cálculos:

1. Se realiza la petición y se obtiene la respuesta
2. Se comprueba que exista el campo `distances`, en el se encuentran las coordenadas
3. Se crea un `MapPoint` con el `id` del `MapPoint` pasado como argumento y las coordenadas x,y,z del primer elemento que aparece en `distances`

A continuación a parece la función de la clase `ServerFacade.java` que realiza estas tareas:

```
    public void getLastPosition (final OnServerResponse
        <MapPoint> responseListener, final MapPoint point) {

        if (!User.getInstance().isLogged()) {
            throw new IllegalStateException(
                "Cannot retrieve tags without
                    a established connection");
        }

        String url = User.getInstance().getToken().getURL().concat(
            "/api/centers/6/beacons/" + point.getId());

        JsonRequest<JSONObject> request = new JsonObjectRequest(
            Request.Method.GET, url, null
                , new Response.Listener<JSONObject>() {
```

```
            @Override
            public void onResponse(JSONObject response) {
                try {
                    final JSONArray distances =
                        response.getJSONArray("distances");

                    if(distances==null
                        || distances.length()==0){
                            return;
                    }

                    final MapPoint newPosition =
                        new MapPoint(
                            point.getId(),
                            (float)distances.getJSONObject(
                                0).getDouble("x"),
                            (float)distances.getJSONObject(
                                0).getDouble("y"),
                            distances.getJSONObject(
                                0).getInt("z"));
                    responseListener.onSuccess(newPosition);

                } catch (JSONException e) {
                    e.printStackTrace();
                    responseListener.onError(Errors.INVALID_JSON);
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Errors errorNum = getServerError(error);
                responseListener.onError(errorNum);
            }
        }) {
        @Override
        public Map<String, String> getHeaders()
            throws AuthFailureError {
                Map<String, String> headers = new HashMap<>();
                headers.put("Authorization", "Bearer " +
                    User.getInstance().getToken().getToken());
                return headers;
            }
        };
        requestQueue.add(request);
    }
```

6.6. Nueva funcionalidad: Registro de nuevos usuarios

Para implementar esta funcionalidad se ha seguido la misma metodología que se usa para iniciar sesión. Con la diferencia de que ahora en la petición REST que se envía al servidor hay que incluir el centro, no sólo el usuario y contraseña, como sucedía en el inicio de sesión.

Se ha creado la actividad `RegisterActivity.java` con su correspondiente *layout*. En la que se incluye un formulario para rellenar los datos de nuevo registro.

A la clase `user.java` se ha añadido la función `register()` similar a la función `login()`, de esa misma clase. Con la diferencia de que en lugar de llamar a la función `getToken()` de la clase `ServerFacade.java`, se llama a la función `register()` de esa misma clase.

Por último, se ha añadido la función `register()` a la clase `ServerFacade.java`, similar a `getToken()`, que como cambio más significativo tiene la URL a la que se realiza la petición, ya que el se ha tenido que implementar un nuevo *end-point* en la API, al ser los datos que se envían en ambas funciones diferentes.

A continuación aparece el fragmento de código dónde se realiza lo mencionado anteriormente:

```
public void register(
    final OnServerResponse<JSONObject> responseListener
        , final String url, final String username
            , final String password, final String center) {

        final String apiURL =
            url.concat("/api/mobileAuthentication?type=register");

        JSONObject params = new JSONObject();

        try {
            params.put("login", username);
            params.put("password", password);
            params.put("center", center);
        } catch (JSONException e) {
            ...
        }
        JsonRequest<JSONObject> request
            = new JsonObjectRequest(apiURL , params , new
                Response.Listener<JSONObject>(){
                    ...
        }
        ...
}
```

Después de describir todos los cambios realizados en el siguiente capítulo se explicarán los detalles sobre la implementación de la aplicación.

Capítulo 7

Implementación

Este capítulo trata sobre cómo se ha realizado la fase de implementación, una vez analizado el diseño de la aplicación. Se explica las cuestiones más relevantes sobre el entorno de desarrollo y las soluciones dadas a las diferentes cuestiones presentadas en el Capítulo 5.

7.1. Entorno de desarrollo

Para el desarrollo de la aplicación se ha utilizado Andoid Studio, entorno de desarrollo oficial de Google [14] y *Java Runtime Environment (JRE)*, necesario para la ejecución de aplicaciones Java [15], ya que en el momento en el que se desarrolló la aplicación original, Kotlin todavía no era el lenguaje de referencia de Google para el desarrollo de sus aplicaciones [14]. Se ha utilizado como sistema de automatización de la compilación *Gradle* [16].

7.2. Versiones necesarias de software

Skywalker está desarrollado para Android 5.0 (Lollipop) o superior.

7.3. Aspectos relativos al código

El código sigue los siguientes criterios de diseño:

- Todos los elementos con carácter literal o estático se han definido como constantes.
- Los hilos serán clases internas a los controladores, para así mejorar el rendimiento y no realizar llamadas del tipo `getAtributo()`.
- Se creará una copia de atributos como posiciones u orientaciones para evitar inconsistencias en los datos.
- El objetivo es obtener el máximo rendimiento, no se ha tenido en cuenta el impacto sobre la batería.

7.4. Eventos en las actividades y fragmentos

Toda aplicación Android necesita tratar los eventos que se generan en sus actividades y fragmentos, ignorarlos puede provocar consumo excesivo de la batería, errores en su estado o su detención completa.

Para el desarrollo de la App se han aprovechado los eventos de parada(`onStop()`, `onPause()`) para la detención de los sensores, la cámara, el envío de paquetes *Bluetooth* y los hilos de dibujado y de peticiones; y los eventos de continuación(`onStart()`, `onResume()`) para reactivarlos.

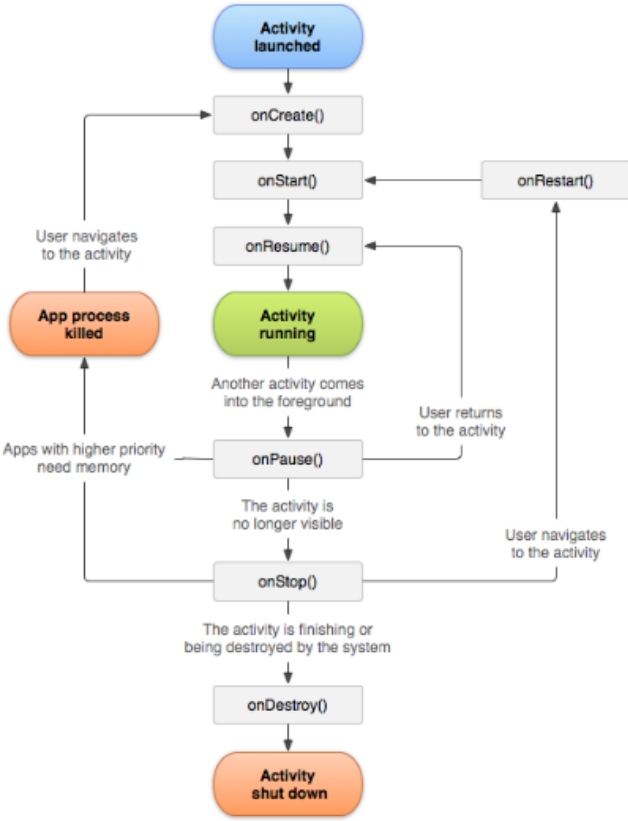


Figura 7.1: Ciclo de vida de una actividad

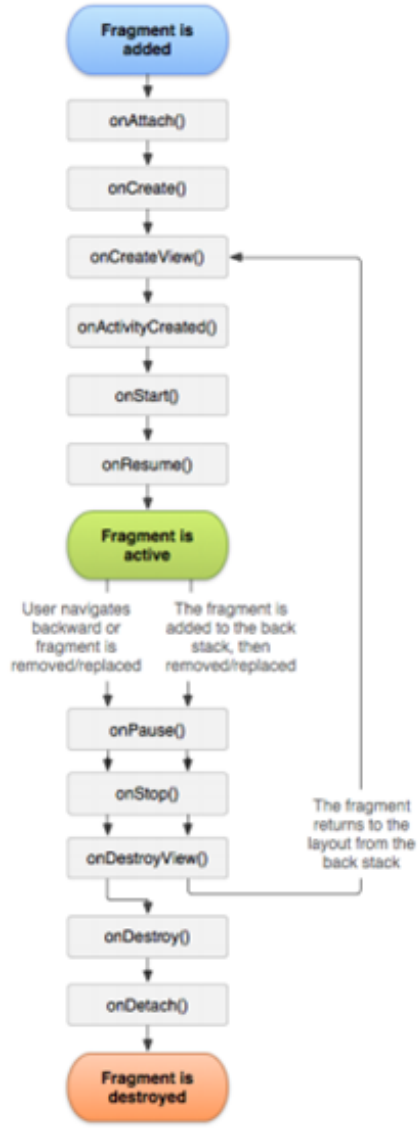


Figura 7.2: Ciclo de vida de un fragmento

La actividad de realidad aumentada no sigue el flujo anteriormente descrito debido a que se producía un cierto retraso al girar el dispositivo ya que cuándo esto sucede la actividad se destruye y tiene que volverse a crear. El ciclo de vida de la actividad es el siguiente:

1. Creación de la vista.
2. Apertura de la cámara.
3. Inicio de una sesión en la cámara.
4. Cambio en la orientación.
5. Detención de la sesión de la cámara.
6. Configuración de la sesión de la cámara.
7. Inicio de una sesión en la cámara.
8. Regreso al paso 4.

7.5. Códigos QR

La aplicación base, utiliza la API Vision [17] para el escaneo de códigos QR, esta API también se puede utilizar para el reconocimiento de sonrisas en fotografías o textos.

Para poder integrar la cámara con la API, se ha utilizado un hilo extra, el cuál, con cada ciclo de su ejecución, recoge el `bitmap` de lo que la cámara ve en cada momento, crea un marco nuevo, y se lo pasa a la API para su procesamiento. Esto es necesario porque en la interfaz de usuario se utilizan `Texture View's`.

El código QR deberá contener un `JSON` con los siguientes campos:

- `scheme`: su valor debe ser `XtremeLoc`, esto sirve para comprobar que el código sigue un formato válido.
- `url`: su valor debe ser la URL del servidor dónde se conectará la App.
- `username`: su valor coincidirá con el nombre de usuario.
- `password`: su valor será el de la contraseña utilizada para iniciar sesión.

Por lo que el JSON que formará el QR será similar al siguiente:

```
{   "scheme":"XtremeLoc",
    "url":"https://xtremeloc.nombreExplotacion.com",
    "username":"user",
    "password":"pass"
}
```

7.6. Peticiones a *Xtremloc*

7.6.1. Biblioteca *Volley*

Se ha utilizado la biblioteca externa *Volley* [18] para las peticiones al servidor. Es la recomendada por Google y tiene las siguientes características:

- Habilita el uso de caché para peticiones vía red.
- Concurrencia transparente para el programador.
- Soporte para peticiones JSON.
- Facilidad de uso.

7.6.2. Peticiones `GET` y `POST`

Xtremeloc solicita un *token* de autenticación para responder a las peticiones, por lo que se han tenido que utilizar cabeceras de peticiones HTTP personalizadas incluyendo el siguiente campo:

```
Autorithation: Bearer 1234Token
```

Como puede observarse en el Diagrama de operaciones REST(Figura 5.28), para el desarrollo de la aplicación sólo se utilizan los métodos `GET` y `POST`:

- `GET`: para solicitar información sobre las antenas y los beacons.
- `POST`: para el envío de información del *token* o del registro del nuevo usuario.

7.7. Señal *iBeacon*

Como se explicó en la Sección 6.4, se utiliza un `ByteBuffer` [13] con el formato *ibeacon* [6], junto con un *Callback*, para el envío de señales *Bluetooth*. El resto de operaciones se realizan utilizando las herramientas proporcionadas por la librería *open-source* `Android Beacon Library` [19].

7.8. Compatibilidad del dispositivo y eventos de conexiones

Se ha mantenido el mecanismo de detección de compatibilidad al iniciar la App, debido a que algunos usuarios no tienen en cuenta los requisitos mínimos para instalar la aplicación. Si el dispositivo no fuera compatible, se avisa al usuario y la aplicación finaliza.

También se ha conservado la activación automática del *Bluetooth* al inicio de la ejecución de la aplicación, y un mecanismo de confirmación de desconexión durante su funcionamiento, para evitar que se cierre la aplicación por un descuido. Estos eventos de conexión y desconexión se gestionan mediante `Broadcast Listener` [20].

Para la conexión con el servidor se utilizan los errores devueltos por `Volley` [18] para detectar desconexiones, informar al usuario y volver a la pantalla de inicio de sesión.

7.9. Precisión del sensor de orientación

Android permite 4 niveles de precisión para el sensor de orientación:

1. Desconocido
2. Bajo
3. Medio
4. Alto

Se ha mantenido el mismo criterio que en la aplicación original: si *Skywalker* detecta que el nivel no es el más alto, se muestra un dialogo al usuario con instrucciones de cómo calibrar su dispositivo. Cuando el sensor detecte que ha sido calibrado, y haya llamado al *Callback* con precisión alta, este dialogo se ocultará.

7.10. Control de versiones

Para el control de versiones se ha utilizado la herramienta Git, en concreto un repositorio privado gestionado por GitLab. Proporciona un entorno gráfico mediante una página web, en el que se pueden ver las diferentes versiones del proyecto. Esto ha permitido cambiar de equipo de trabajo durante el desarrollo del proyecto.

Después de describir la fase de implementación, en el siguiente capítulo se exponen las diferentes pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.

Capítulo 8

Plan de pruebas y evaluación

En este capítulo se detallan las pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación y el cumplimiento de los requisito descritos en el Capítulo 2 y los casos de uso recogidos en el Capítulo 4

8.1. Pruebas sobre autenticación y registro

En esta sección se detallan las pruebas realizadas para comprobar la correcta creación de usuarios y posterior conexión con el servidor.

Prueba 1.1	
Descripción	Conexión con el servidor
Acción	Se envía una petición a *Xtremeloc* con el usuario y contraseña.
Resultado esperado	Se inicia sesión en la aplicación y se comienza a emitir señales *Bluetooth* en formato *iBeacon* y se reciben las posiciones de los demás elementos.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-01

Tabla 8.1: Prueba 1.1 - Conexión con el servidor

Prueba 1.2	
Descripción	Creación de usuario
Acción	Se envían los datos a *Xtremloc* para su almacenamiento y se realiza la conexión con el servidor.
Resultado esperado	Se inicia sesión en la aplicación y se comienza a emitir señales *Bluetooth* en formato *iBeacon* y se reciben las posiciones de los demás elementos.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-08

Tabla 8.2: Prueba 1.2 - Creación de usuario

8.2. Pruebas sobre filtrado de elementos

En esta sección se detallan las pruebas destinadas a comprobar el correcto funcionamiento sobre el filtrado de puntos de interés.

Prueba 2.1	
Descripción	Selección de nuevos elementos
Acción	La aplicación muestra un listado con todos los puntos de interés del centro y permite seleccionarlos o deseleccionarlos.
Resultado esperado	La aplicación muestra sobre la visión de la cámara trasera los nuevos puntos de interés.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-02

Tabla 8.3: Prueba 2.1 - Selección de nuevos elementos

8.3. Pruebas sobre visionado de elementos

En esta sección se detallan las pruebas realizadas para comprobar el correcto visionado de los puntos de interés.

Prueba 3.1	
Descripción	Dibujado de los puntos de interés
Acción	La aplicación dibuja los puntos de interés seleccionados que se encuentran dentro de la visión de la cámara trasera, indicando su distancia. Aquéllos que que no se encuentren dentro de su visión los dibuja mediante una flecha indicando su dirección.
Resultado esperado	Visionado de los puntos seleccionados incluyendo su distancia, si se encuentran dentro del ángulo de visión de la cámara trasera y de la dirección de aquellos que no se encuentran dentro de su ángulo de visión.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-03, RF-06

Tabla 8.4: Prueba 3.1 - Dibujado de puntos de interés

Prueba 3.2	
Descripción	Actualización de los puntos de interés
Acción	La aplicación muestra las nuevas distancias u orientaciones según se desplaza el usuario.
Resultado esperado	Modificación de la distancia según nos acercamos o alejamos a un punto de interés o incluir la distancia de un punto cuándo éste entra dentro del ángulo de visión de la cámara trasera del dispositivo.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-03, RF-04, RF-05, RF-06

Tabla 8.5: Prueba 3.2 - Actualización de puntos de interés

Prueba 3.3	
Descripción	Cambio de orientación del dispositivo
Acción	El usuario cambia la orientación del dispositivo o ésta se cambia accidentalmente.
Resultado esperado	Los puntos de interés que se están mostrando se recolocan sobre la nueva visión de la cámara trasera.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-03, RF-06

Tabla 8.6: Prueba 3.3 - Cambio de orientación del dispositivo

8.4. Pruebas sobre cierre de sesión

En esta sección se detallan las pruebas destinadas a comprobar el correcto funcionamiento del cierre de sesión.

Prueba 4.1	
Descripción	Cierre de sesión
Acción	El usuario cierra sesión desde la vista de realidad aumentada
Resultado esperado	La vista de realidad aumentada se destruye, la aplicación conexión con el servidor se interrumpe y se muestra la vista de nueva conexión
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RF-07

Tabla 8.7: Prueba 4.1 - Cierre de sesión

8.5. Pruebas sobre interfaz de usuario

En esta sección se detallan las pruebas realizadas para comprobar el correcto funcionamiento de cada elemento de la interfaz gráfica.

Prueba 5.1	
Descripción	Idioma
Acción	Navegación por todas las vistas con el dispositivo configurado en castellano e inglés.
Resultado esperado	Todos los elementos de la interfaz de usuario aparecen en su idioma correspondiente.
Resultado obtenido	Coincide con el esperado.

Tabla 8.8: Prueba 5.1 - Idioma

Prueba 6.2	
Descripción	Botón inicio sesión en vista tutorial
Acción	En la vista de tutorial se pulsa el botón iniciar sesión.
Resultado esperado	La aplicación muestra la vista con el formulario para realizar la conexión con el servidor.
Resultado obtenido	Coincide con el esperado.

Tabla 8.9: Prueba 6.2 - Vista tutorial

Prueba 6.3	
Descripción	Cambio de pestañas en vista iniciar sesión/ realizar conexión
Acción	Se pincha en cada uno de los títulos de las pestañas: Entrada manual, Conexion QR, Demostración.
Resultado esperado	Se muestra la pestaña correspondiente.
Resultado obtenido	Coincide con el esperado.

Tabla 8.10: Prueba 6.3 - Cambio de pestaña

Prueba 6.4	
Descripción	Botón registrar nuevo usuario en vista inicio sesión/realizar conexión
Acción	Se pulsa sobre el botón Registrar nuevo usuario de la vista inicio sesión/realizar conexión.
Resultado esperado	La aplicación muestra la vista Registro nuevo usuario con el formulario correspondiente.
Resultado obtenido	Coincide con el esperado.

Tabla 8.11: Prueba 6.4 - Botón Registrar usuario

Prueba 6.5	
Descripción	Número máximo de puntos de interés seleccionados
Acción	El usuario tiene seleccionados el número máximo de puntos de interés seleccionados e intenta añadir otro.
Resultado esperado	La aplicación no permite añadir este último punto a la lista.
Resultado obtenido	Coincide con el esperado.

Tabla 8.12: Prueba 6.5 - Número máximo de puntos de interés seleccionados

8.6. Tratamiento de errores

En esta sección se detallan las pruebas destinadas a comprobar el correcto funcionamiento de la aplicación cuándo aparece un error.

Prueba 6.1	
Descripción	Inicio de sesión con un dispositivo que no cumple los requisitos mínimos
Acción	Se inicia la aplicación en un dispositivo que carece de los sensores necesarios o de la versión mínima de Android.
Resultado esperado	Se muestra un diálogo informando al usuario de la incompatibilidad y se cierra la aplicación.
Resultado obtenido	Coincide con el esperado.
Requisitos satisfechos	RNF-01

Tabla 8.13: Prueba 6.1 - Dispositivo no cumple requisitos mínimos

Prueba 6.2	
Descripción	URL incorrecta
Acción	Se introduce una URL incorrecta al realizar la conexión o crear un nuevo usuario.
Resultado esperado	La aplicación muestra un mensaje informando al usuario
Resultado obtenido	Coincide con el esperado.

Tabla 8.14: Prueba 6.2 - URL incorrecta

Prueba 6.3	
Descripción	Credenciales incorrectas
Acción	Las credenciales introducidas para realizar la conexión son incorrectas
Resultado esperado	La aplicación muestra un mensaje informado al usuario
Resultado obtenido	Coincide con el esperado.

Tabla 8.15: Prueba 6.3 - Credenciales incorrectas

Prueba 6.4	
Descripción	Desactivar el *Bluetooth* con la aplicación en funcionamiento
Acción	Se desactiva el *Bluetooth* con la aplicación en funcionamiento.
Resultado esperado	La aplicación muestra un mensaje dando al usuario la opción de cerrar sesión o activar el *Bluetooth*.
Resultado obtenido	Coincide con el esperado.

Tabla 8.16: Prueba 6.4 - *Bluetooth* desactivado con App funcionando

Prueba 6.5	
Descripción	Error de red
Acción	No hay conexión a Internet durante el inicio de sesión.
Resultado esperado	La aplicación muestra un mensaje informando al usuario
Resultado obtenido	Coincide con el esperado.

Tabla 8.17: Prueba 6.5 - Error de red

Prueba 6.6	
Descripción	Inicio de sesión con *Bluetooth* apagado
Acción	Se inicia sesión con el *Bluetooth* apagado.
Resultado esperado	La aplicación activa el *Bluetooth* e inicia sesión.
Resultado obtenido	Coincide con el esperado.

Tabla 8.18: Prueba 6.6 - Inicio de sesión con *Bluetooth* apagado

Prueba 6.7	
Descripción	Error en la conexión con el servidor
Acción	Se produce algún error en la conexión con el servidor durante el funcionamiento de la visión de realidad aumentada.
Resultado esperado	La aplicación informa al usuario y muestra la vista de inicio de sesión/realizar conexión.
Resultado obtenido	Coincide con el esperado.

Tabla 8.19: Prueba 6.7 - Error en la conexión

En el siguiente capítulo se describen, en el manual de programador, aquellos aspectos de la aplicación que tienen que tener en cuenta los desarrolladores que tengan que modificarla o añadir funcionalidades en el futuro.

Capítulo 9

Manual de programador

En este capítulo se detallada la información más relevante que un desarrollador debe tener en cuenta a la hora de modificar o ampliar el funcionamiento de *Skywalker*

9.1. Material

- **Equipo de desarrollo**, pudiendo ser Linux, Windows o Mac.
- **Dispositivo Android**, con sistema operativo Android 5.0 o superior.
- **Android Studio**, en su última versión. La máquina de desarrollo debe cumplir una serie de requisitos mínimos [14]:
  - Sistema operativo
    - Linux: GNOME o KDE
    - Windows: Windows 7 o superior
    - Mac: Mac OS X 10.10 (Yosemite) o superior
  - Memoria: 4GB de memoria RAM
  - Disco: 2GB de espacio disponible en disco

  Junto a Android Studio se instalan los JDK y SDK.
- **Postman o aplicación similar**, aplicación para realizar peticiones a API's que facilitará las pruebas.
- **XtremeLoc**, instalado en el lugar de desarrollo para poder comprobar el funcionamiento de la aplicación.
- **Emisor BLE** para comprobar el correcto funcionamiento de la aplicación.

9.2. Estructura y funcionamiento de la aplicación

Al abrir el proyecto con Android Studio, se observa que consta de de cuatro carpetas:

- **Business**, en ella se encuentran cada una de las clases que componen la lógica del negocio de la aplicación. Cabe destacar que la clase *User* utiliza el patrón *Singleton*.
- **Persistence**, en ella se encuentra la clase *S*erverFacade, que utiliza una patrón *Singleton*, en las que se realizan todas las peticiones a la API de XtremeLoc.
- **Presentation**, en ella están agrupadas todas las actividades de las que se compone la aplicación.
- **Services**, que agrupa las clases con utilidades de estilo matemático como vectores o matrices, necesarias para el funcionamiento de la aplicación.

En el capítulo siguiente, manual de usuario, se describe cómo tiene que utilizar la aplicación el usuario final.

Capítulo 10

Manual de usuario

En este capítulo se detalla el funcionamiento de la aplicación desde el punto de vista del usuario final.

10.1. Acceso a la aplicación

Existen dos métodos para acceder a la aplicación. Si ya se tiene un usuario creado, se debe iniciar sesión. Si por el contrario es la primera vez que se usa la aplicación se debe realizar un nuevo registro.

10.1.1. Iniciar sesión

Se puede iniciar sesión de dos maneras diferentes: utilizando credenciales, usuario y contraseña, o mediante un código QR.

Mediante credenciales

Se deben rellenar los campos que aparecen a continuación: el servidor al que se debe conectar el usuario y sus credenciales personales. Después se debe pulsar sobre el botón *Aceptar*

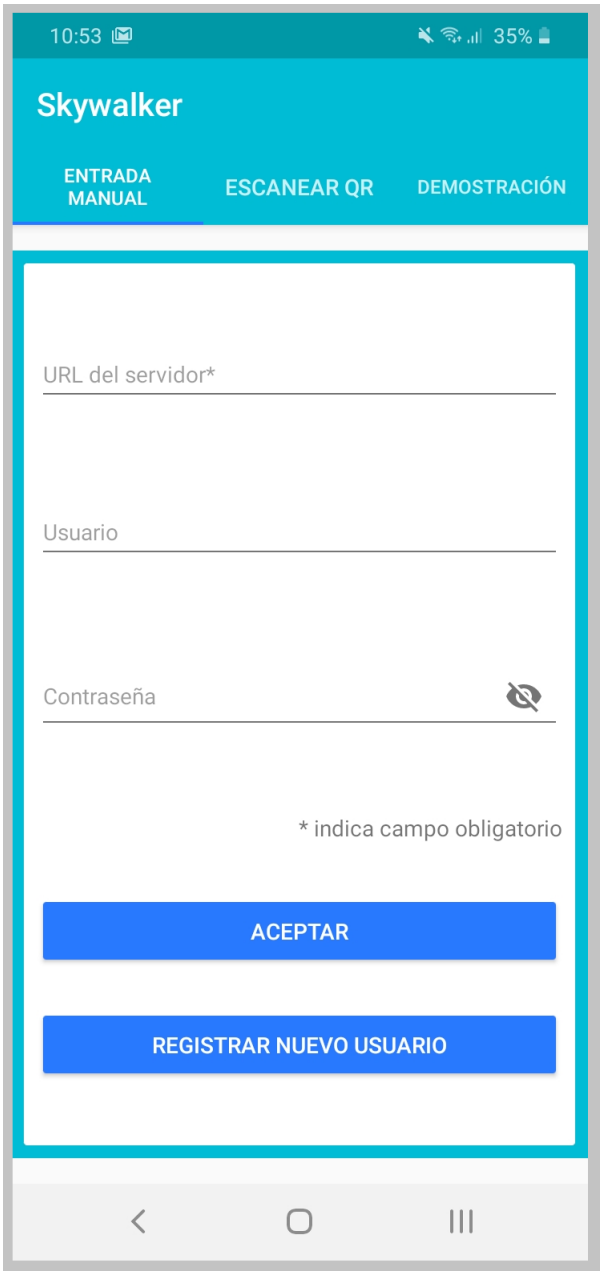


Figura 10.1: Inicio de sesión mediante credenciales

Mediante código QR

Si se quiere iniciar sesión mediante un código QR, el usuario debe pulsar sobre *Escanear QR* del menú superior de la imagen anterior y colocar la cámara trasera sobre dicho código para que el dispositivo sea capaz de reconocerlo.

Figura 10.2: Inicio de sesión mediante código QR

10.1.2. Nuevo Registro

Si es la primera vez que el usuario utiliza la aplicación, primero se deberá registrar, para ello el usuario deberá pulsar sobre botón *Registrar nuevo usuario* de la Figura 10.1 y completar el formulario que aparece en siguiente imagen.

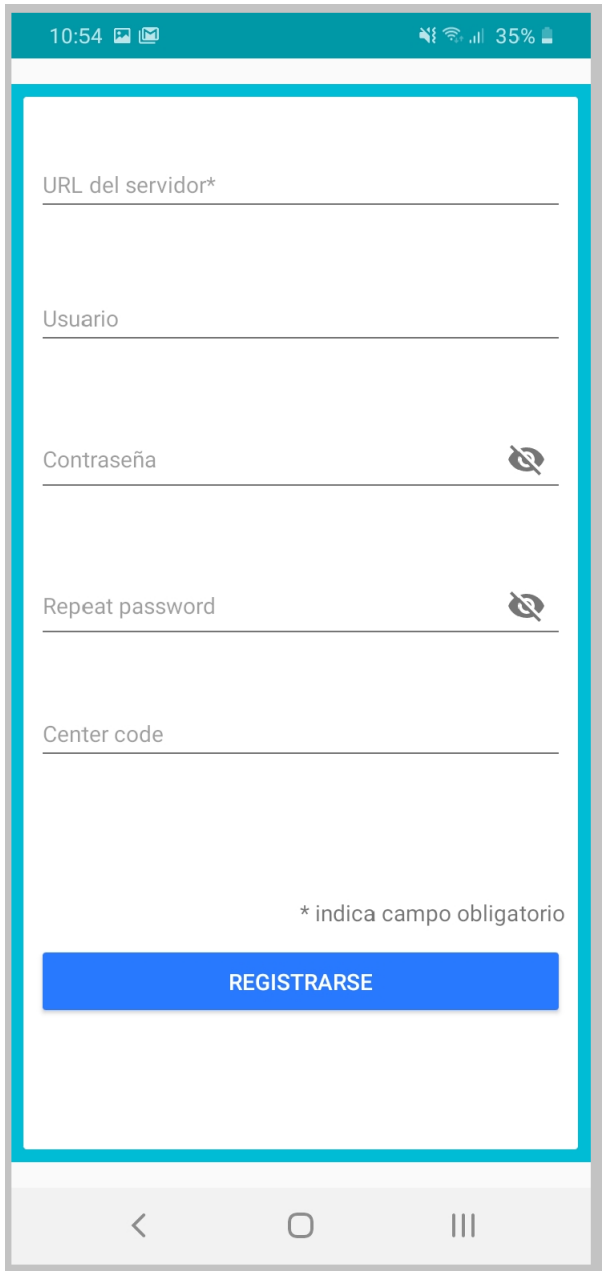


Figura 10.3: Nuevo Registro

El usuario deberá conocer la URL del servidor a la que la aplicación debe conectarse y el código del centro en el que usará la aplicación. El nombre de usuario deberá ser único, no debe coincidir con el nombre de ningún otro usuario. También se le pedirá la confirmación de la contraseña.

Una vez completado los campos se debe pulsar sobre el botón *Registrarse*

10.2. Filtrar elementos

Por defecto, al iniciar sesión aparecerán los primeros cinco elementos que se detectan, si les hubiese.

Si el usuario no está interesado en todos ellos o el/los elemento/s en el/los que está interesado no aparece/n deberá buscarlo/s. Para ello el usuario tiene que desplegar el menú lateral pulsando sobre la pantalla del dispositivo y seleccionar *Filtrar elementos*

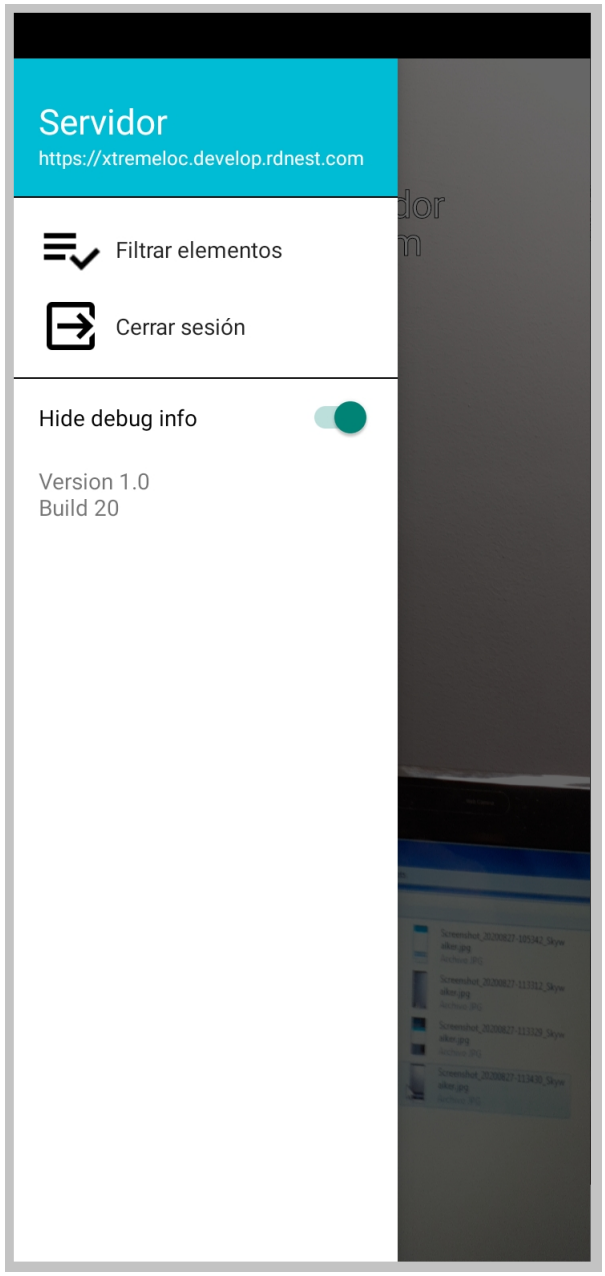


Figura 10.4: Menú Lateral

Aparecerá la lista de elementos que puede buscar, pudiendo seleccionar sólo aquéllos que sean de su interés. Una vez realizada la selección se debe pulsar sobre el botón *Aceptar*

Figura 10.5: Elementos seleccionables

10.3. Modo demostración

La aplicación cuenta con un modo demostración para poderla enseñar en entornos en los que *Xtremeloc* todavía no se encuentra instalado.

Para acceder a este modo el usuario debe pulsar sobre *Modo demostración* del menú superior de la Figura 10.1

Se pide al usuario que confirme el acceso al modo demostración, para ello debe pulsar sobre el botón *Modo demostración*. Un vez el usuario acceda al modo demostración el funcionamiento es igual, salvo que las distancias no cambian al no estar instalado *Xtremeloc* y los elementos que aparecen son simulados.

Figura 10.6: Modo Demostración

Por último, en el siguiente capítulo se expondrán la conclusiones a las que se ha llegado realizando este TFG.

Capítulo 11

Conclusiones y trabajo futuro

En este capítulo se describirá el trabajo realizado a lo largo del TFG, enumerando las funcionalidades implementadas una vez finalizado el proyecto.

11.1. Conclusiones

Para la realización de este TFG se ha realizado un estudio de la aplicación *Skywalker* y del sitema de localización en interiores *Xtremloc* para poder volver a poner en funcionamiento la aplicación móvil. A continuación se resume el trabajo realizado

- Se ha realizado un estudio en profundidad de la API de *Xtremloc* y de la aplicación *Skywalker*
- Se ha comprendido el funcionamiento del protocolo *iBeacon*, para la transmisión de paquetes *Bluetooth*.
- Se ha modificado el sistema de permisos en el servidor.
- Se ha desarrollado el sistema de registro de nuevo usuario.
- Se ha modificado el tratamiento de las respuesta de la API de *Xtremeloc*, ya que estas habían cambiado y los cambios no habían sido tenidos en cuenta en la aplicación.
- Se ha conseguido que la distancia a la que la distancia que aparece en la App cambie según se desplaza el dispositivo, el *beacon* o ambos, y que tenga un margen error asumible.
- Se ha documentado cada una de las operaciones REST que realiza la aplicación a la API, para que si en el futuro ésta se modifica, se pueda comprobar si estos cambios afectan a la aplicación.
- Se han realizado los manuales de usuario y programador de la aplicación.
- Se ha hecho un exhaustivo plan de pruebas para comprobar el correcto funcionamiento de la aplicación.

11.2. Trabajo Futuro

La principal funcionalidad que se puede añadir a esta aplicación, es la de crear puntos de interés desde la propia aplicación, estos serían fijos y coincidirían con las coordenadas del dispositivo en el momento en el que se crea el punto.

En este momento sólo podrían crearse de manera local, se almacenarían en el dispositivo, ya que *Xtremloc* no permite, todavía, que sean los dispositivos los que creen elementos de búsqueda.

Esta funcionalidad puede ser de gran utilidad, por ejemplo, para marcar la ubicación de una obra en un museo o exposición o localizar una sala de reuniones, oficina o despacho, dónde ya se haya estado, dentro de un edificio.

Referencias

[1] Victor Rojo. *Reeingeniería y desarrollo de un gestor de actualizaciones software para localización indoor*. 2019

[2] Antonio Román. *Desarrollo de front-end para administración de componentes de un sistema de localización indoor*. 2019

[3] Pablo Renero Balgañón. *Integración de tecnología Ultra-wideband en un sistema de localización indoor basado en Bluetooth*. Julio 2020

[4] Iván González Rincón *Diseño y desarrollo de un servicio de gestión de planos de edificios basado en OpenStreetMaps*. Septiembre 2020

[5] Maria D. Miñambres, Victor Rojo, Antonio Roman, Pablo Renero, and Diego R. Llanos. *Versatile, Low-cost Indoor Positioning Combining Bluetootht and Ultra Wideband Technologies.*ICL-GNSS (International Conference on Localization and GNSS), Tampere, Finland, june 2020. ISSN 1613-0073

[6] Formato trama iBeacon. *https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/*. Marzo 2015. Último acceso 03/08/2020

[7] Buildroot. `https://en.wikipedia.org/wiki/Buildroot`. Junio 2020. Último acceso 03/08/2020

[8] Craig Larman. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Pearson educación, 2003.

[9] Fragmentos Android. `https://developer.android.com/guide/components/fragments.html`, Diciembre 2019. Último acceso 03/08/2020

[10] Hilos Android. `https://developer.android.com/guide/components/processes-and-threads.html`, Julio 2020. Último acceso 03/08/2020

[11] Matrices de rotación. `https://en.wikipedia.org/wiki/Rotation_matrix`, Septiembre 2020. Último acceso 02/09/2020

[12] Ángulo de visón de la cámara. `https://en.wikipedia.org/wiki/Angle_of_view`, Junio 2020. Último acceso 04/08/2020

[13] ByteBuffer Java. `https://docs.oracle.com/javase/7/docs/api/java/nio/ByteBuffer.html`. Último acceso 20/05/2020

[14] Android Studio. https://en.wikipedia.org/wiki/Android_Studio, Junio 2020. Último acceso 13/07/2020.

[15] Jesús Tomás Gironés. *El gran libro de Android*. Marcombo, 2018.

[16] Gardle. https://gradle.org/features/. Último acceso 04/08/2020.

[17] API Vision. https://developers.google.com/vision/. Último acceso 04/08/2020

[18] Volley Framework. https://developer.android.com/training/volley/index.html, Junio 2020. Último acceso 04/08/2020.

[19] Android Beacon Library. https://altbeacon.github.io/android-beacon-library/. Último acceso 04/08/2020.

[20] Broadcast Listener. https://developer.android.com/guide/components/broadcasts.html, Diciembre 2019. Último acceso 04/08/2020