UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Monitorización y control de los módulos de un jardín inteligente

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Laia Ferrando Ferragud

Tutor: Poza Luján, José Luis
Posadas Yagüe, Juan Luis

2015-2016

Resumen

En los últimos años han aparecido una gran variedad de sensores y actuadores. Para estudiar cómo incorporar estos avances tecnológicos a los entornos abiertos, surge el sistema del que este proyecto forma parte.

El principal objetivo de este proyecto es ofrecer al usuario una aplicación móvil para plataforma Android que le permita una supervisión y un control sobre los elementos de jardinería de un entorno distribuido.

Para ello, se han realizado los análisis oportunos tanto de mercado, como de características y funcionalidades. El diseño y funciones finales han surgido de las conclusiones sacadas de dichos análisis.

Palabras clave: Aplicaciones móviles, entornos inteligentes, Android, Arduino, servicio web REST, Java

Abstract

Recently a wide variety of sensors and actuators have appeared. To study how to incorporate these technological advances to open environments, the system of which this project it arises.

The main objective of this project is to provide the user with an Android platform application that allows control and supervision over the gardening elements of a distributed environment.

To this end, the appropriate analysis have both been made: market, features and functionalities. The final design and functions have emerged from the conclusions drawn from these analysis.

Keywords : Mobile applications, intelligent environments, Android, Arduino, REST web services, Java

Tabla de contenidos

Ilustraciones

Tablas

1 Introducción

1.1 Entorno y Motivación

Actualmente los sistemas de automatización de procesos en entornos abiertos, como jardines o espacios públicos, suelen estar dedicados a tareas básicas como el control del riego, y controlados por unidades centralizadas. Sin embargo, gracias a los avances tecnológicos de los últimos años, es posible automatizar más procesos, a partir de la medición de variables ambientales (humedad, presión atmosférica, etc.) y por medio de una mayor cantidad de actuadores (riego selectivo o iluminación ambiental). Para estudiar cómo incorporar estos avances tecnológicos a los entornos abiertos, surge el sistema del que este proyecto forma parte.

Dado que en un jardín o espacio público, se puede disponer de una gran cantidad de sensores[1] y actuadores[2] por planta o conjunto de plantas, se plantea la posibilidad de distribuir parte de la automatización y de proporcionar la información de los sensores o el control de los actuadores a usuarios que se conecten, a través de Internet, al sistema por medio de ordenadores o de dispositivos móviles. Para ello, es necesario disponer de aplicaciones móviles que se conecten al sistema y den el acceso antes mencionado.

1.2 Objetivos del proyecto

A partir de lo que se ha expuesto anteriormente, se plantea el objetivo principal de este proyecto que es: ofrecer al usuario una aplicación móvil que le permita una supervisión y un control sobre los elementos de jardinería de un entorno distribuido. Las condiciones que intervienen en el crecimiento de una planta deberán ser monitorizadas de forma remota, así como la oportunidad de intervenir en el proceso de riego o en el cambio de ubicación (en el caso de macetas motorizadas) desde la propia aplicación.

1.3 Estructura del documento

Este documento está compuesto de 6 secciones. A continuación se describen brevemente cada una de ellas:

En la primera sección se hace una introducción al contenido del proyecto, sus objetivos y el entorno en el que se realiza.

En la segunda sección se realiza un estudio de mercado, para tener una visión global de las herramientas disponibles y sus características. A partir de ello, se extraen las características que debe tener esta aplicación.

En la tercera sección se define de forma técnica y más detallada todos los aspectos que debe ofrecer la aplicación, así como la descripción global del proyecto en su totalidad(todas las partes implicadas).

En la cuarta sección se presenta el diseño del sistema, la distribución de los todos los datos y la forma de interacción elegida.

En la quinta sección se detallan las implementaciones a las que se ha optado de acuerdo al diseño seleccionado.

Finalmente se exponen las conclusiones de este proyecto, así como las dificultades encontradas y las soluciones a éstas.

2 Entorno

2.1 Introducción

A continuación se va a introducir dentro de qué campo o campos se realiza el proyecto, además de realizar una exploración de las aplicaciones que puedan ofrecer unos servicios similares. Para finalizar, a partir de las características que ofrecen otras aplicaciones, se extraerán las propias al proyecto, adaptándose a las necesidades del sistema.

2.2 Entorno de realización

El cuidado de una planta o cultivo requiere tiempo, pero sobre todo necesita un continuo control. Solo con una condición inadecuada para el vegetal puede ralentizar o incluso matar el cultivo.

En el sector industrial ya es posible mantener este control del entorno, incluso se pueden automatizar condiciones de ambiente que no son propias de la zona para conseguir cultivos extranjeros.

Sin embargo en el ámbito particular es más difícil encontrar sistemas que permitan ejercer este tipo de control, ya que las herramientas existentes en el mercado son algo escasas y en algunos casos incompletas.

En este contexto surge el sector de la ingeniería informática, ya que ofrece la posibilidad de desarrollar un completo sistema de control y automatización para un entorno particular.

2.3 Sistemas similares

Se ha hecho un estudio de mercado cuyo objetivo es encontrar aplicaciones con unas características similares a las que se busca para la realización de este proyecto, así como decidir qué aspecto puede ayudar a esta aplicación a destacar entre las demás. Se ha hecho un análisis de las mismas, incluyendo para qué sistema operativo está disponible(Android[3], iOS...)

2.3.1 EDYN

Edyn es un sistema que consta de un accesorio que se inserta en la tierra junto a tus plantas. Este sensor recoge diferentes datos sobre el clima y la tierra. La aplicación muestra estos datos en tiempo real, además de enviar alertas y sugerencias para el completo desarrollo de tus plantas. Está disponible para dispositivos Android y iOS.

Ilustración 1. Interfaz de la aplicación EDYN[1]

2.3.2 Loxone Smart Home

Loxone Smart Home es una aplicación de control de domótica. Esta aplicación incluye el control de las persianas automáticas, la temperatura dentro y fuera del hogar, gestor de energía, etc. Además, este sistema ofrece estadísticas en tiempo real del consumo de energía.

Está disponible para dispositivos Android y iOS.

(Nota: Esta aplicación no se incluirá en la posterior tabla comparativa, ya que aunque tiene unas características de diseño y funcionalidad similares al sistema que se desea desarrollar, no pertenece al campo de la jardinería.)

2.3.3 Green IQ

GreenIQ va conectado al GreenIQ Smart Garden Hub. Este accesorio, junto con la aplicación, sirve para programar el riego de las diferentes zonas que se delimiten. De esta forma se ahorra agua, a la vez que se puede controlar el estado de las plantas. Esta aplicación está más orientada al ahorro de agua que a la automatización del cultivo. Únicamente disponible para dispositivos Android.

[1]fuente: https://edyn.com/

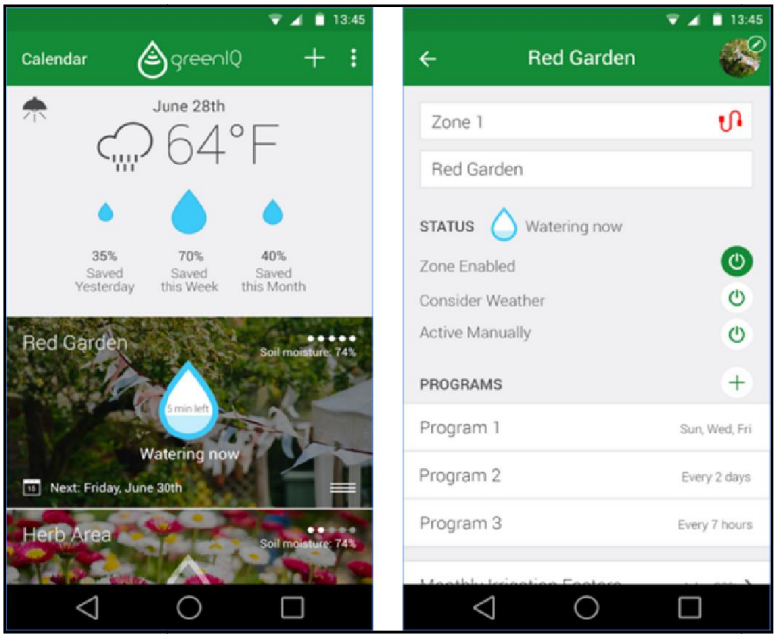

Ilustración 2.Interfaz de la aplicación GreenIQ[2].

MEG

Ofrece niveles tanto de temperatura como de luz, necesidad de abono y depósito de agua. Además tiene incorporada en la base de la maceta una luz que se puede configurar para emitir el color deseado. Parece ser sólo un prototipo, ya que no está disponible para descarga.

Ilustración 3. Interfaz de la aplicación MEG[3].

[2] Fuente: http://greeniq.co/
[3] Fuente: http://www.gizmag.com/internet-connected-greenhouse-smartphone-meg/34727/

2.3.4 ParrotPot

La aplicación de ParrotPot va enlazada a una maceta que integra su propio depósito y sistema de riego. Dicha maceta programa los riegos según el tipo de planta que se le indique. Además, monitoriza los niveles de humedad, temperatura, fertilidad e iluminación.

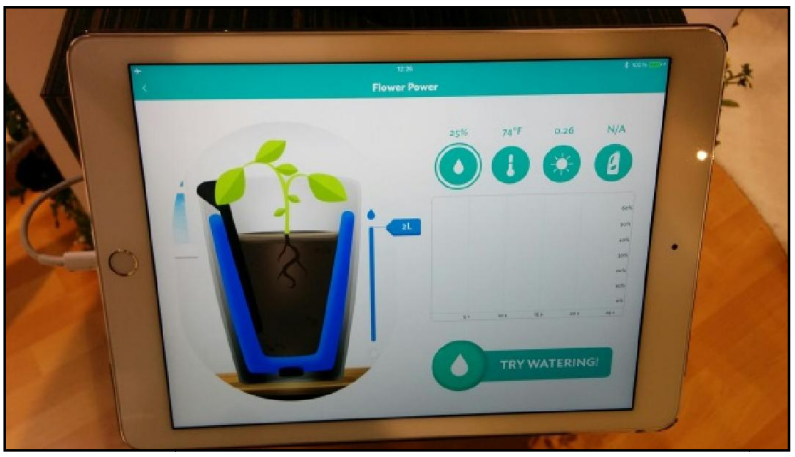

Ilustración 4. Interfaz aplicación ParrotPot[4].

2.3.5 ParrotFlowerPower

Este sistema proporciona monitorización sobre una planta con la ayuda de un accesorio que se inserta junto a la planta(en la tierra). Este accesorio recoge datos sobre temperatura, humedad, cantidad de luz y lo envía a la aplicación.

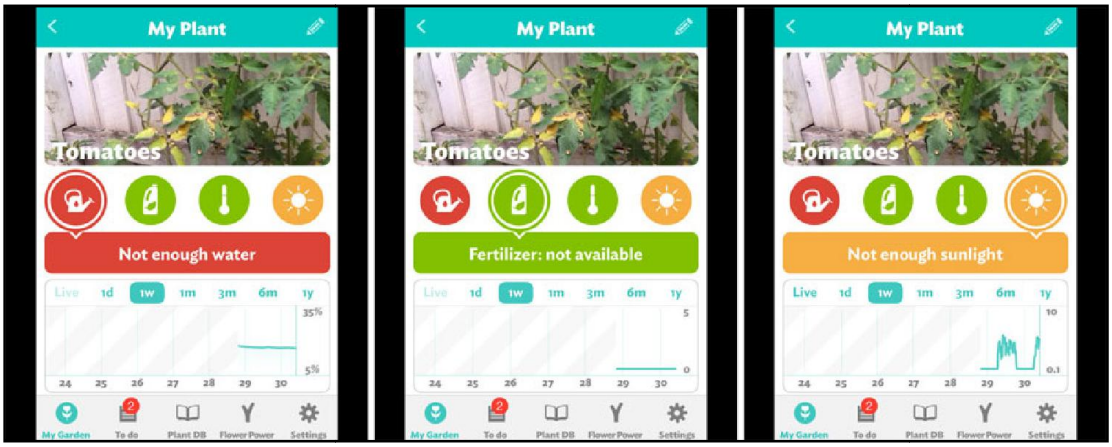

Ilustración 5.Interfaz de la aplicación ParrotFlowerPower[5].

[4] Fuente: http://www.parrot.com/ces/?_ga=1.140616306.150116909.1465585491
[5] Fuente: http://www.androidworld.it/2014/12/05/recensione-parrot-flower-power-261903/

2.4 Análisis

En este subcapítulo se quiere hacer una comparación de diversas características para obtener por un lado las funciones básicas que deberá tener la aplicación, y por otro lado la o las características que harán que la aplicación destaque entre su competencia. A continuación se muestra una tabla comparativa con diferentes características:

T: Temperatura. Describe si se da soporte a la consulta de temperatura. Sí/No.

NDep: Nivel de depósito. Describe sise da soporte a la consulta del nivel del depósito de agua. Sí/No.

NHum: Nivel de humedad. Describe si se da soporte a la consulta del porcentaje de humedad en el ambiente. Sí/No.

Il: Iluminación. Describe si se da soporte a la consulta de el nivel de iluminación. Sí/No.

US: Ultrasonido. Describe si se da soporte a la consulta de la distancia a la cual se encuentra un obstáculo mediante un sensor de ultrasonidos. Sí/No.

Mov: Movimiento. Describe si se da soporte a la capacidad para indicar a la maceta que cambie su posición. Sí/No.

R. Aut: Riego automático. Describe si se da soporte a la capacidad de activar y desactivar el riego automático. Sí/No.

S.O: Sistema operativo. Indica para cuál o cuáles sistemas operativos está disponible la aplicación.

Hist: Historial. Describe si se da soporte a la consulta del historial de datos de uno o más sensores.

Sistema	T.	NDep	NHum	Il	US	Mov.	R. Aut.	S.O.	Hist.
2.3.1	Sí	No	Sí	Sí	No	No	No	Android,iOS	Sí
2.3.3	Sí	No	Sí	No	No	No	Sí*	Android	Sí
2.3.4	Sí	No	Sí	Sí	No	No	No	Ninguno	No
2.3.5	Sí	Sí	Sí	Sí	No	No	Sí	Android	Sí
2.3.6	Sí	No	Sí	Sí	No	No	No	Android,iOS	Sí

Tabla 1. Comparación de características de las aplicaciones[6]

*Es necesario de accesorios externos para esta funcionalidad.

2.5 Síntesis

Completado el análisis de las distintas aplicaciones existentes, se procede a resumir y extraer las características que se desean conseguir en la aplicación. En dichas características se utilizará el código CaXX como numeración.

Ca01: Capacidad de mostrar valores de sensores en tiempo real. Los sensores incluyen control de temperatura, humedad, luz, nivel de depósito de agua y ultrasonidos.

Ca02: Posibilidad de mostrar el registro de valores de los sensores.

Ca03: Control sobre el desplazamiento de la maceta.

Ca04: Capacidad para detener la maceta si ésta está en movimiento.

Ca05: Posibilidad de activar y desactivar el riego automático.

Ca06: Posibilidad de definir distintos tipos de usuarios según los cuales se pueda controlar que características mostrar y cuáles restringir.

Ca07: Mostrar el historial de valores de un sensor en particular.

Toda esta información debe ser recibida o enviada a un servidor REST que se encargue de transmitirla a la maceta o macetas en cuestión.

[6]Fuente: propia

2.6 Tecnología a emplear

El siguiente paso es decidir sobre qué plataforma se va a desarrollar el proyecto, y la Tabla 1. Comparación de características de las aplicaciones da una idea de cuál puede ser la opción más adecuada. Se puede apreciar que, pese a que algunas de las aplicaciones están disponibles para más de una plataforma, en todas las que se encuentran al alcance de los usuarios prima la plataforma Android. Por ello y también por cuestiones de disponibilidad de recursos se ha optado por esta plataforma.

Para el desarrollo de aplicaciones Android es bastante común utilizar el kit SDK y además existen diferentes herramientas sobre las que programar. En este caso se ha optado por Android Studio[4], un entorno ya conocido que proporciona unas buenas características y es muy útil para ciertos aspectos de la programación. Además incluye un emulador en el que se pueden hacer todo tipo de pruebas sobre la aplicación.

2.7 Conclusiones

En esta sección se ha hecho un repaso de las aplicaciones referentes a cultivo automatizado que existen en el mercado. A pesar de que hay bastantes en la actualidad son mejorables, ya que se pueden completar aún más ofreciendo un par de funcionalidades bastante interesantes como es el movimiento remoto o el sensor de ultrasonidos asociado a éste.

Sin duda una aplicación enlazada a un sistema como es el del cultivo automatizado ayuda al usuario a tener una mayor interacción con él, y también permite tener toda la información fácilmente accesible.

Terminado el análisis comparativo se han extraído las características que sería idóneo que el sistema tuviese. En general queremos que la aplicación sea capaz de mostrar toda la información de forma clara y ordenada, además de permitirnos controlar algunos aspectos como son el riego o el movimiento.

Por último se ha optado por la plataforma Android, junto con el programa Android Studio, que nos permitirá realizar toda la implementación de forma ordenada y sencilla.

3 Especificación de requisitos

Este apartado se incluye para presentar formalmente las funcionalidades y requisitos que definen el proyecto. Para ello se ha elegido el estándar IEE 830.

Como añadido se proporcionará información sobre el equipo que conforma en su totalidad el proyecto.

3.1 Terminología

En este aparto se dispone a esclarecer algunos términos que puedan resultar confusos o puedan tener un significado diferente dentro del proyecto:

Maceta: Cuando se nombra una maceta se refiere a una unidad que contiene un microcontrolador Arduino, una serie de sensores(temperatura, humedad, luz, ultrasonidos y nivel de agua), un depósito de agua y diversos actuadores(bomba de riego y dos motores para el movimiento de la maceta en sí).

Sistema: Se refiere al proyecto global que incluye a todo el grupo de personas que se encargan de hacer sus correspondientes partes de éste.

Proyecto: Hace referencia a la sección del sistema dedicada al diseño e implementación de una aplicación móvil, detallada en este documento.

3.2 Funcionalidad

3.2.1 Equipo involucrado y perspectiva del producto

Este apartado muestra la información de todas las personas involucradas en el proyecto así como una descripción del sistema en general.

Nombre	Axel Guzmán Godia
Rol	Desarrollador automatización.
Categoría profesional	Estudiante
Responsabilidades	Desarrollar el código Arduino y montar el Hardware
Información de contacto	axguzgo@inf.upv.es

Tabla 2. Miembro Axel Guzmán Godia

Nombre	Laia Ferrando Ferragud
Rol	Desarrolladora Aplicación móvil.
Categoría profesional	Estudiante
Responsabilidades	Diseño aplicación móvil
Información de contacto	laferfer@inf.upv.es

Tabla 3. Miembro Laia Ferrando Ferragud

Nombre	Israel Beltrán Arias
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones entre módulos.
Información de contacto	isbelar@inf.upv.es

Tabla 4. Miembro Israel Beltrán Arias

Nombre	Alejandro Delgado
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones módulo-servidor
Información de contacto	-

Tabla 5. Miembro Alejandro Delgado

Nombre	José Luis Poza Luján
Rol	Supervisor.
Categoría profesional	Profesor Contratado Doctor
Responsabilidades	Supervisar proyecto.
Información de contacto	jopolu@disca.upv.es

Tabla 6. Miembro José Luis Poza Luján.

Nombre	Juan Luis Posadas Yagüe
Rol	Supervisor.
Categoría profesional	ProfesorTitular
Responsabilidades	Supervisarproyecto.
Información de contacto	jposadas@disca.upv.es

Tabla 7. Miembro Juan Luis Posadas Yagüe

Una vez detallada la información referente al equipo que compone el sistema, se pasa a detallar cómo está distribuido éste. El sistema está dividido en diferentes partes.

La primera es la orientada al montaje de una maceta con diferentes elementos hardware, además de la programación del microcontrolador Arduino para la automatización de su funcionamiento.

La segunda está centrada en la comunicación entre macetas(protocolo) así como en la comunicación desde una maceta al servidor.

La tercera es la dirigida a gestionar por un lado la base de datos en la que se guardará la información de sensores y actuadores, y por otra parte implementar el servidor que se encargará de comunicarse con la aplicación móvil y con las macetas mediante REST.

La cuarta, descrita en este documento, es la formada por el diseño y la implementación de una aplicación móvil para monitorizar los datos de las macetas ytener cierto control sobre ellas.

3.3 Casos de uso

En estos tres casos de uso los usuarios involucrados son el usuario, la aplicación y el servidor, aunque la aplicación se asumirá que va incluida en el Usuario para simplificar los diagramas.

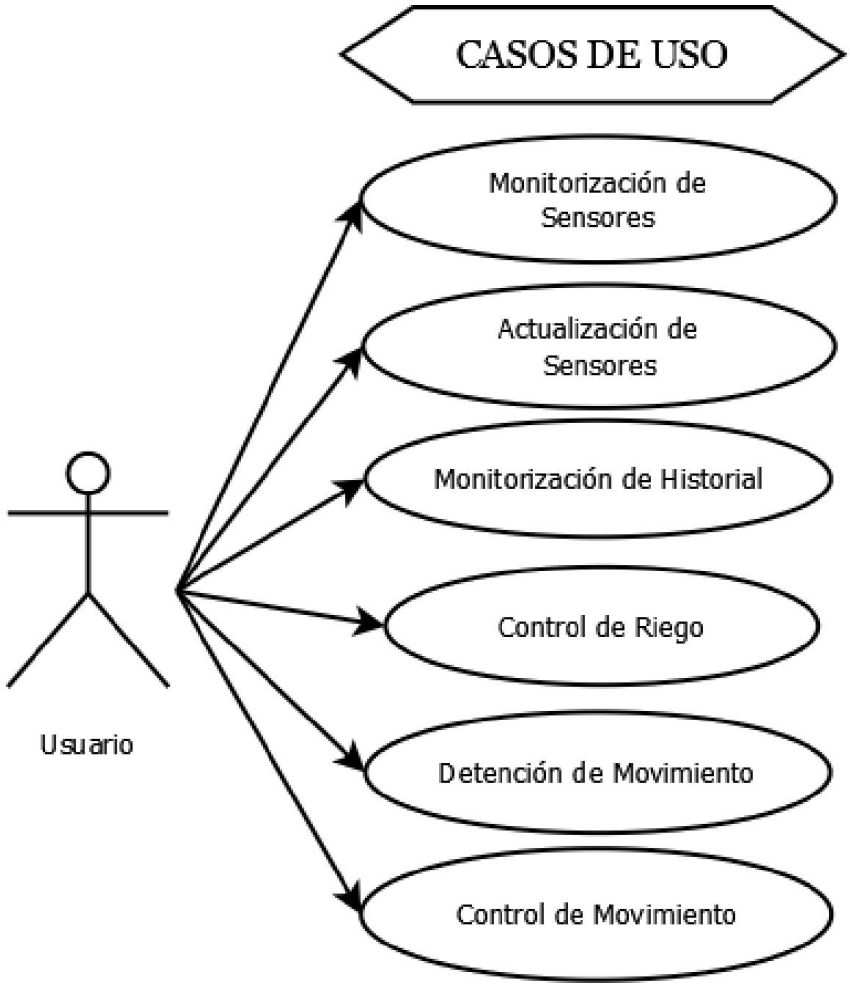

Ilustración 6. Casos de uso

Los casos de uso son 9: la monitorización de los sensores que pueda tener la maceta junto con su historial, la posibilidad de activar y desactivar el riego automático de una maceta y la capacidad de dirigir una maceta(control de movimiento y detención de movimiento). Cada uno de estos casos de uso se van a detallar a continuación.

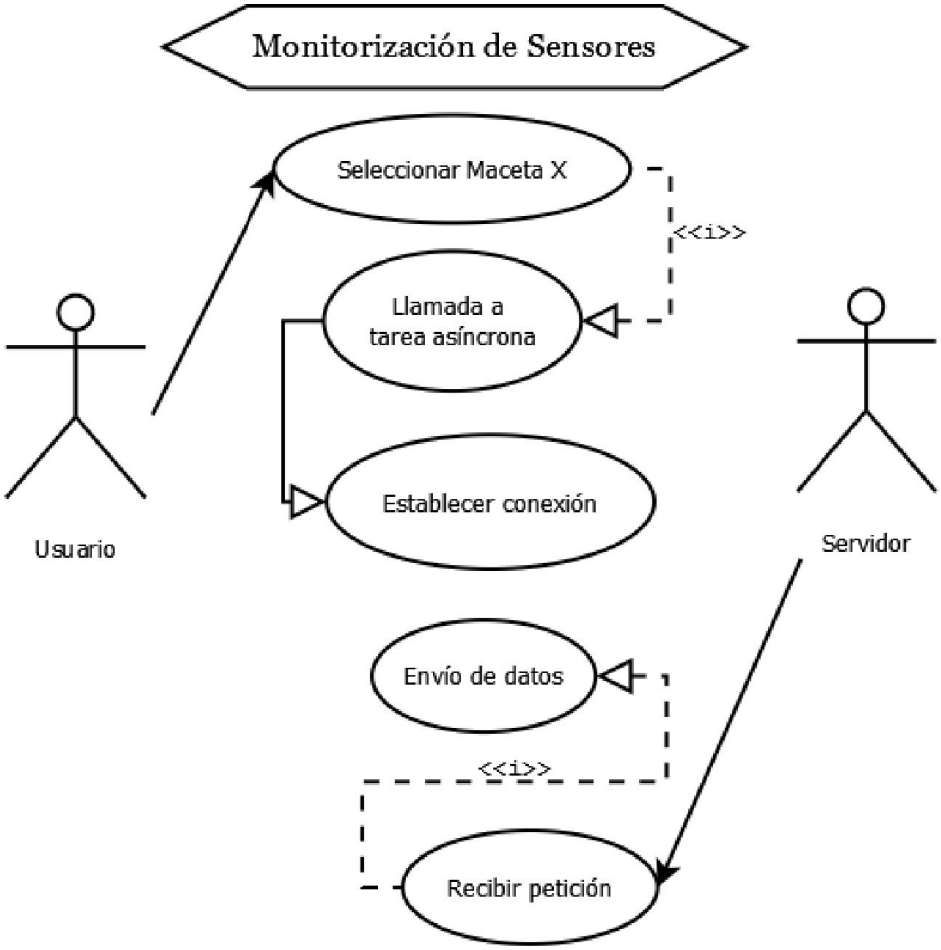

Ilustración 7. Monitorización de sensores

CU_01: Monitorización de sensores. El primer caso de uso es el referente a la obtención de datos por parte del servidor. Aún así, debe ser la aplicación la que solicite dichos datos cuando sean requeridos. Para ello se requiere implementar una tarea asíncrona que establezca una conexión con el servidor, envíe los datos y reciba la información. En el momento de enviar la petición se le debe comunicar al servidor el id de la maceta que hay que monitorizar, ya que se mostrarán todos los sensores de una sola maceta en un determinado momento. Este dato se transmite mediante la misma URL, ya que dicho servidor es REST y está configurado de esta forma.

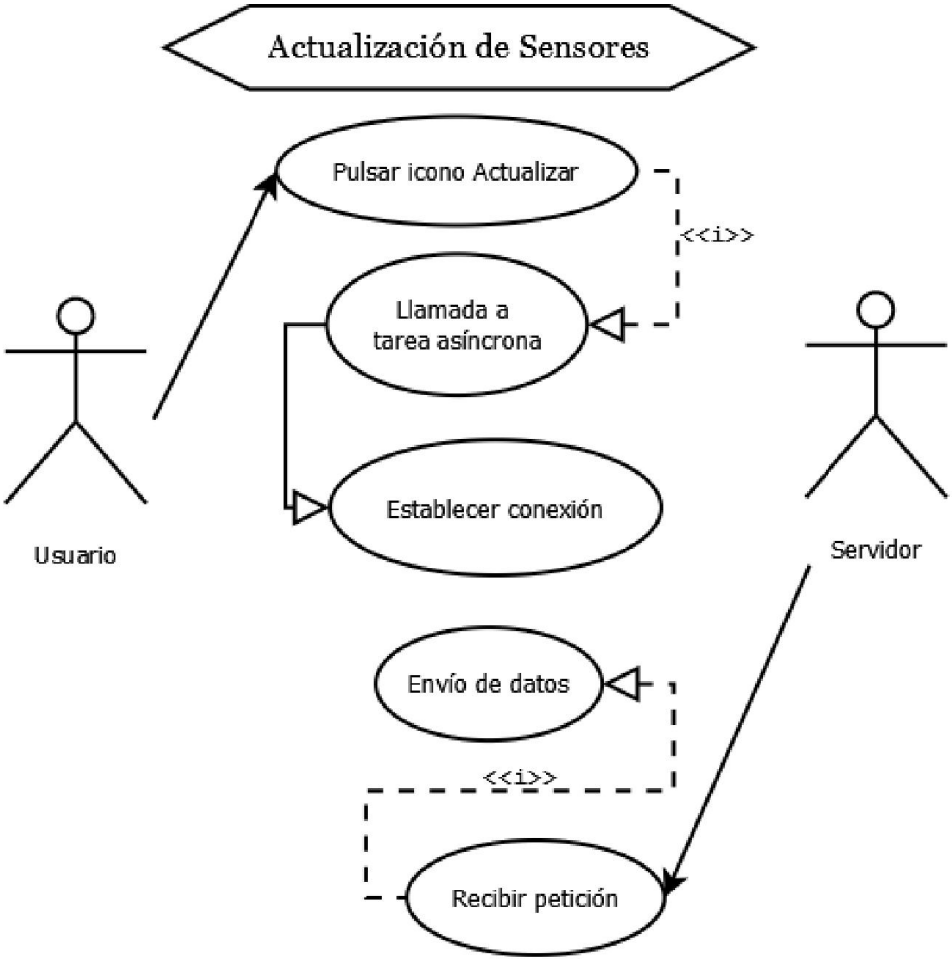

Ilustración 8. Actualización de sensores

CU_02: Actualización de sensores. El segundo caso de uso describe el proceso de actualizar los datos de los sensores que se muestran en la aplicación. Para ello, el usuario debe pulsar el icono de actualizar de la aplicación. Esto origina una llamada a una tarea asíncrona que se encarga de establecer la conexión ya hacer la nueva petición de datos al servidor. A continuación, el servidor proporciona esos datos que se procesan para mostrarlos de nuevo.

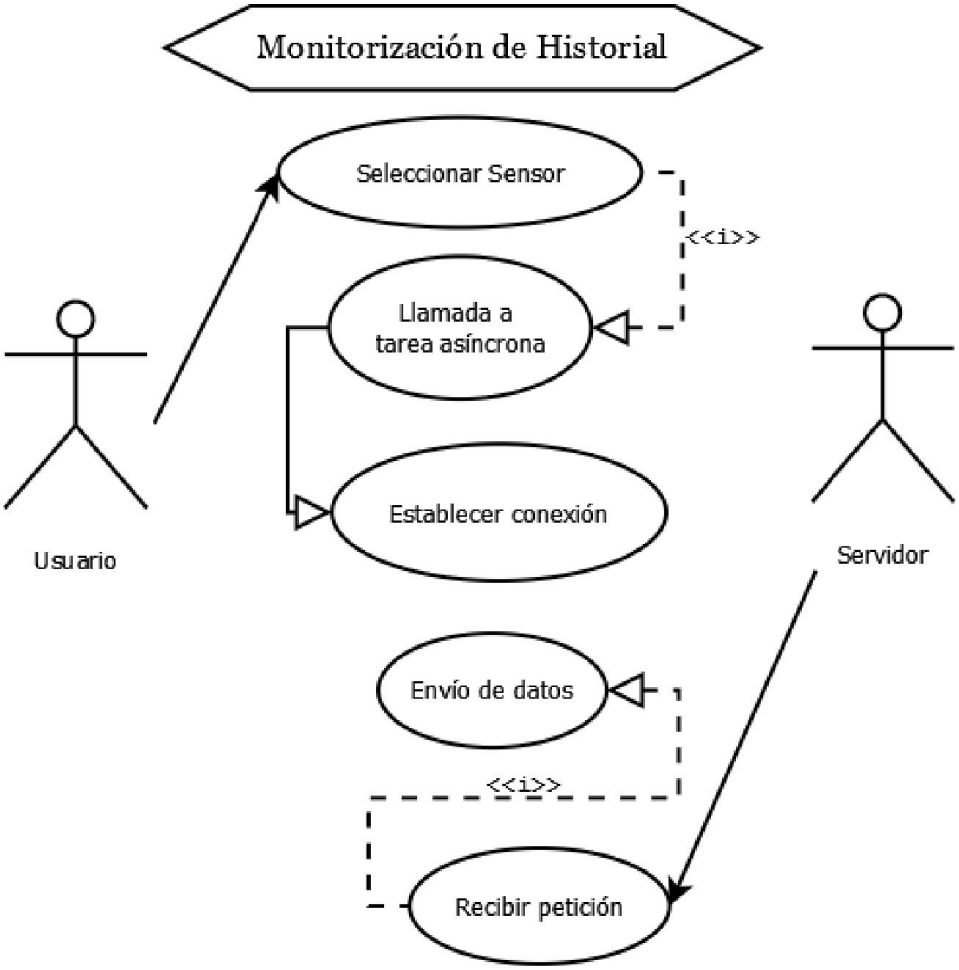

Ilustración 9. Monitorización de Historial

CU_03: Monitorización de Historial. El tercer caso de uso se refiere a la recepción de todos los datos que se tienen de un sensor almacenados en el servidor. Para ello, el usuario debe pulsar sobre el sensor que se quiere consultar, lo que desencadena una llamada a una tarea asíncrona. Esta tarea asíncrona establece conexión con el servidor REST. Por otra parte, cuando se establece la conexión y el servidor recibe la petición, se envían los datos que se piden en base al id de la maceta y el id del sensor.

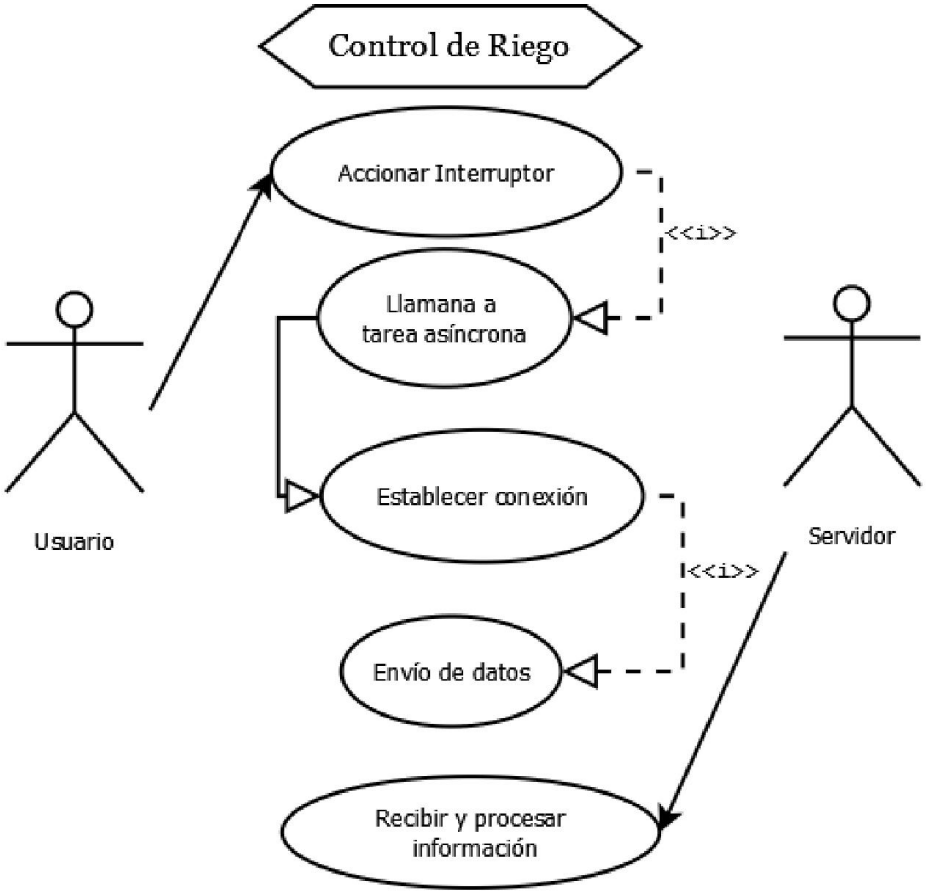

Ilustración 10. Control de Riego

CU_04: Control de Riego. El cuarto caso de uso se asocia con la activación o desactivación del riego. Para ello, el usuario debe activar o desactivar el interruptor (es el mismo para ambos casos). Una vez accionado, se llama a la tarea asíncrona. Dependiendo de si se activa o se desactiva se llamada a la tarea asíncrona se realiza con un valor de entrada u otro. Para la activación se envía un valor 100 y para la desactivación un valor 10. Cuando la tarea tiene dicho dato, se establece conexión con el servidor y se envía el dato. Por último el servidor lo recibe y lo procesa.

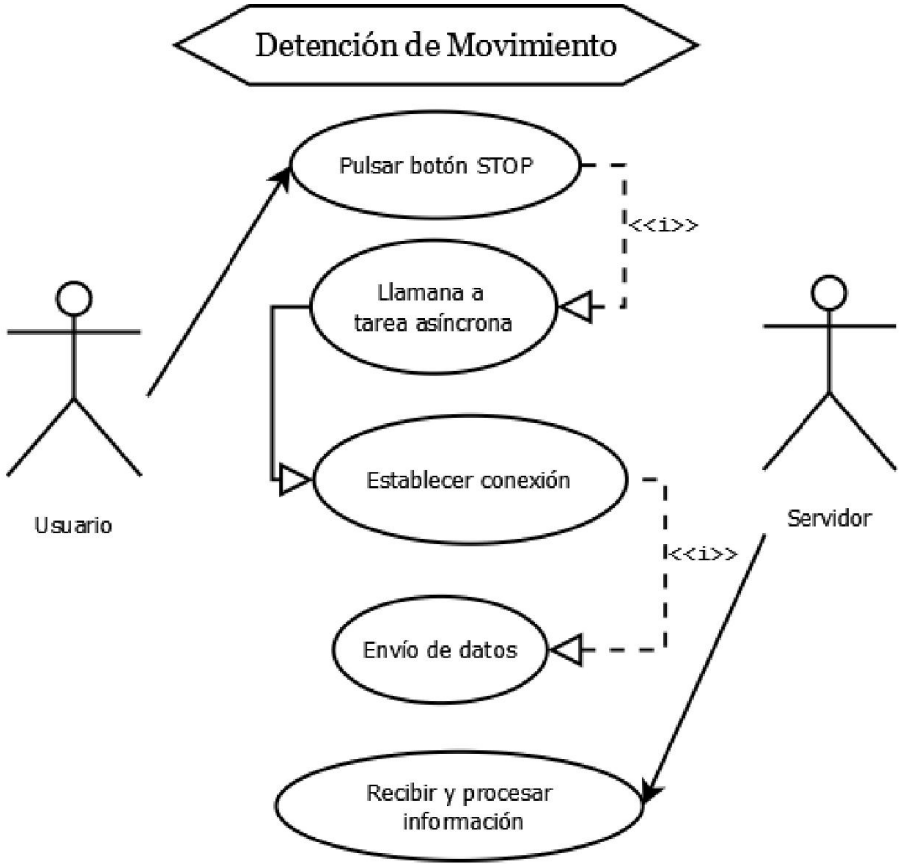

Ilustración 11. Detención de Movimiento

CU_05: Detención de Movimiento. El quinto caso de uso se asocia con la interrupción del movimiento de la maceta. Para ello, el usuario debe pulsar el botón de STOP de la aplicación. Una vez accionado, se llama a la tarea asíncrona. Se envía el valor X= 0 y Y=0 para detener la maceta. Cuando la tarea tiene ambos datos, se establece conexión con el servidor y se envían. Por último el servidor los recibe y los procesa.

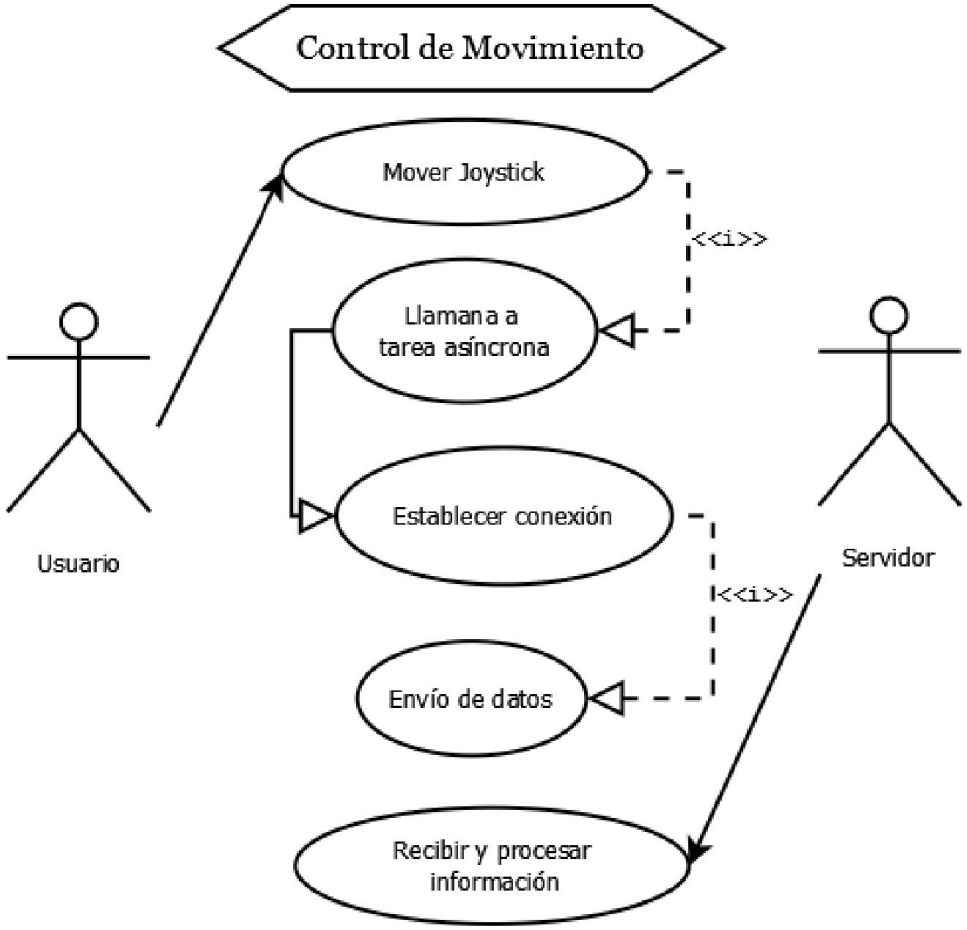

Ilustración 12. Control de Movimiento 1

CU_06: Control de Movimiento 1. El sexto y último caso de uso se asocia con la necesidad del usuario de poder controlar el movimiento de una maceta desde la aplicación. Para ello, el usuario debe deslizar el joystick hacia arriba, abajo, izquierda o derecha. En el momento en que se detecta dicho desplazamiento, se llama a la tarea asíncrona. Se envía el valor X o Y dependiendo de la dirección, junto con el id de la maceta. Cuando la tarea recibe este dato, se establece conexión con el servidor y se envía. Por último el servidor lo recibe y lo procesa.

3.3.1 Tablas de Funcionalidades

Una vez descritos y detallados los casos de uso, hay que describir que funcionalidades deberá cumplir la aplicación. A continuación se muestran una serie de tablas que describen en qué consiste cada funcionalidad y la prioridad de cada una:

Número de Funcionalidad	CU_01
Nombre	Monitorización de sensores
Descripción	Se debe mostrar toda la información que se recoja en la maceta a través de sus sensores, que son: temperatura del aire y del suelo, humedad, luz, ultrasonidos y nivel del depósito.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 8. Funcionalidad 1: Monitorización de sensores

Número de Funcionalidad	CU_02
Nombre	Actualización de sensores
Descripción	Se debe permitir la actualización de los valores actuales de cada uno de los sensores de la maceta.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 9. Funcionalidad 2: Actualización de valores de sensores

Número de Funcionalidad	CU_03
Nombre	Monitorización del historial de sensores
Descripción	Se debe mostrar un listado con todas las mediciones de un determinado sensor, incluyendo la fecha y la hora en la que se ha recogido dicha medición. Deben estar disponibles los historiales de todos los sensores.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 10. Funcionalidad3: Monitorización del historial de sensores

Número de Funcionalidad	CU_04
Nombre	Control de riego
Descripción	Se debe permitir al usuario obtener cierto control sobre el riego, de forma que se pueda activar o desactivar a voluntad desde la aplicación.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 11. Funcionalidad 4: Control de riego

Número de Funcionalidad	CU_05
Nombre	Detención de movimiento
Descripción	Se debe permitir al usuario detener la maceta de forma remota cuando ésta está en movimiento.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 12. Funcionalidad 5: Detención de movimiento

Número de Funcionalidad	CU_06
Nombre	Control de movimiento
Descripción	Se debe permitir al usuario ordenar a la maceta que se mueva desde la aplicación en cualquier momento en que sea posible.
Prioridad	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 13. Funcionalidad 6: Control de movimiento 1(Avance)

3.4 Conclusiones

En este capítulo se han descrito algunos términos que puedan tener un significado distinto fuera de éste. Esto es importante para evitar confusiones a la hora de hablar de ellos.

También se ha hecho un pequeño resumen genérico de las partes que componen este sistema, para así poder ubicar y entender en su totalidad la necesidad de incluir la parte descrita en este documento. Así mismo se han proporcionado los datos de los demás miembros de equipo que lo conforma.

Después se han especificado los casos de uso que surgen en este proyecto, y la forma de abordar cada uno de ellos desde el punto de vista de la implementación.

Por último, a partir de los casos de uso se han extraído los requisitos

En este capítulo se ha podido recopilar todos los datos para definir las funcionalidades del proyecto, así como mostrar de forma genérica como funciona todo el sistema. A partir de los casos de uso definidos y detallados, se especifican los requisitos funcionales que debe cumplir la aplicación.

4 Diseño del sistema

4.1 Introducción

En esta sección se van a mostrar los pasos a seguir para realizar un diseño que sea funcional y fácil de utilizar y que a la vez proporcione todas las opciones que se necesitan para el control de las macetas.

4.2 Especificación conceptual

En esta sección se va a introducir de forma conceptual la estructura que forma todo el sistema. El sistema cuenta con control deliberativo que lo conforman la comunicación entre el servidor y las macetas, así como las macetas entre sí. La función que cumple este proyecto es contribuir con una capa de control inteligente (supervisión por parte del usuario).

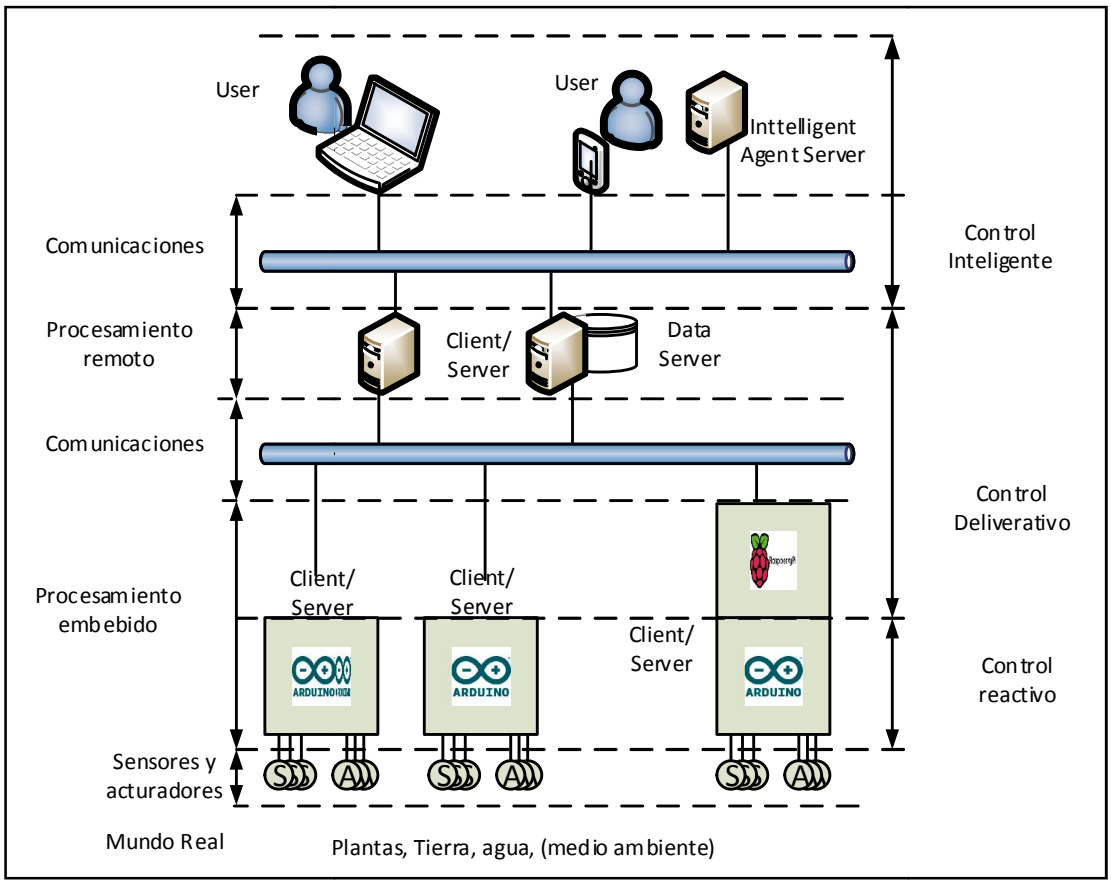

Ilustración 13. Descripción especificación conceptual

4.3 Especificación formal

Para el diseño de la arquitectura se ha optado por el modelo de 3 capas: Persistencia, Negocio y Presentación. Para hacer un buen diseño se debe definir cuidadosamente cada una de las capas. En este caso la capa más importante es la de negocio, pero eso no significa que las demás deban dejarse de lado. A continuación se detallan cada una de las capas de la arquitectura.

4.3.1 Capa persistencia

Para el guardado de los datos se va a hacer uso de la base de datos que se ha configurado junto con el servidor REST por parte de la persona encargada en el proyecto global. Desde ahí, cada vez que se haga una petición al servidor, se recuperarán los datos para mostrarlos en la aplicación.

Por otra parte, para almacenar los datos de forma temporal en la aplicación, se hace uso de las preferencias compartidas(SharedPreferences). Estas preferencias tienen estructura clave-valor, por lo que se puede organizar la información de forma sencilla. En este caso se van a utilizar para guardar en cada momento el identificador(*id*) de la maceta sobre la que se está trabajando.

La monitorización de sensores en particular se debe almacenar de forma más estructurada para mostrar los datos de forma ordenada. Por ello se ha optado por un ArrayList de HashMap cuyo par clave-valor sean String, para poder guardar más variedad de datos sin problema.

4.3.2 Capa negocio / lógica

Las acciones por las que se va a regir la aplicación, descritas en la siguiente ilustración, son las siguientes:

- Esperar la acción del usuario: la mayoría de acciones permitidas en la aplicación se realizan por orden del usuario, por lo que esta parte es la más básica para que el proyecto funcione como es debido.

- Conectar con el servidor: cada vez que se haga una petición, se hace una conexión al servidor REST para obtener o enviar datos. Esta conexión se establece mediante la formación de una URL. La estructura de la URL es proporcionada por la persona encargada de configurar el servidor REST.

- Recepción de Datos (Sensores): al recibir datos, ya sean para el historial de un sensor o para el valor actual de todos los sensores, se debe procesar la información recibida. Para procesar esta información, se debe guardar de forma ordenada en un ArrayList. Este ArrayList está formado por HasMap para organizar bien la información.

- Envío de Información (Riego/Movimiento): según el valor que se quiera transmitir a la maceta se debe almacenar en un array unos valores u otros. En el caso de la activación o desactivación del riego, debe almacenarse un 10 o un 100. Para el caso del movimiento se debe enviar el valor de la Y(100 o -100) para avanzar o retroceder, o el valor de la X(0,100) para girar a izquierda o derecha.

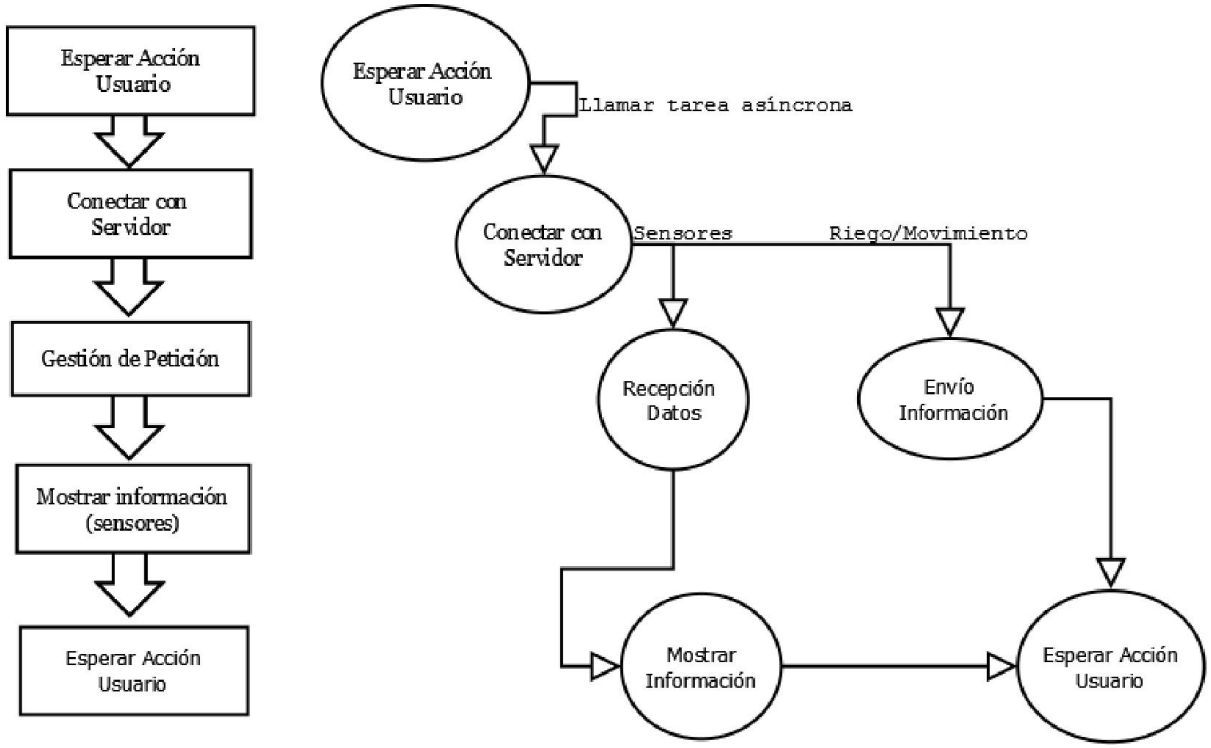

Ilustración 14. Diagrama de Actividad

A continuación se van a presentar y describir los diagramas de secuencia que componen este proyecto. Estos diagramas surgen por la necesidad de especificar cuál o cuáles son los procesos de comunicación entre la aplicación y el servidor. A continuación se detalla cada uno de ellos:

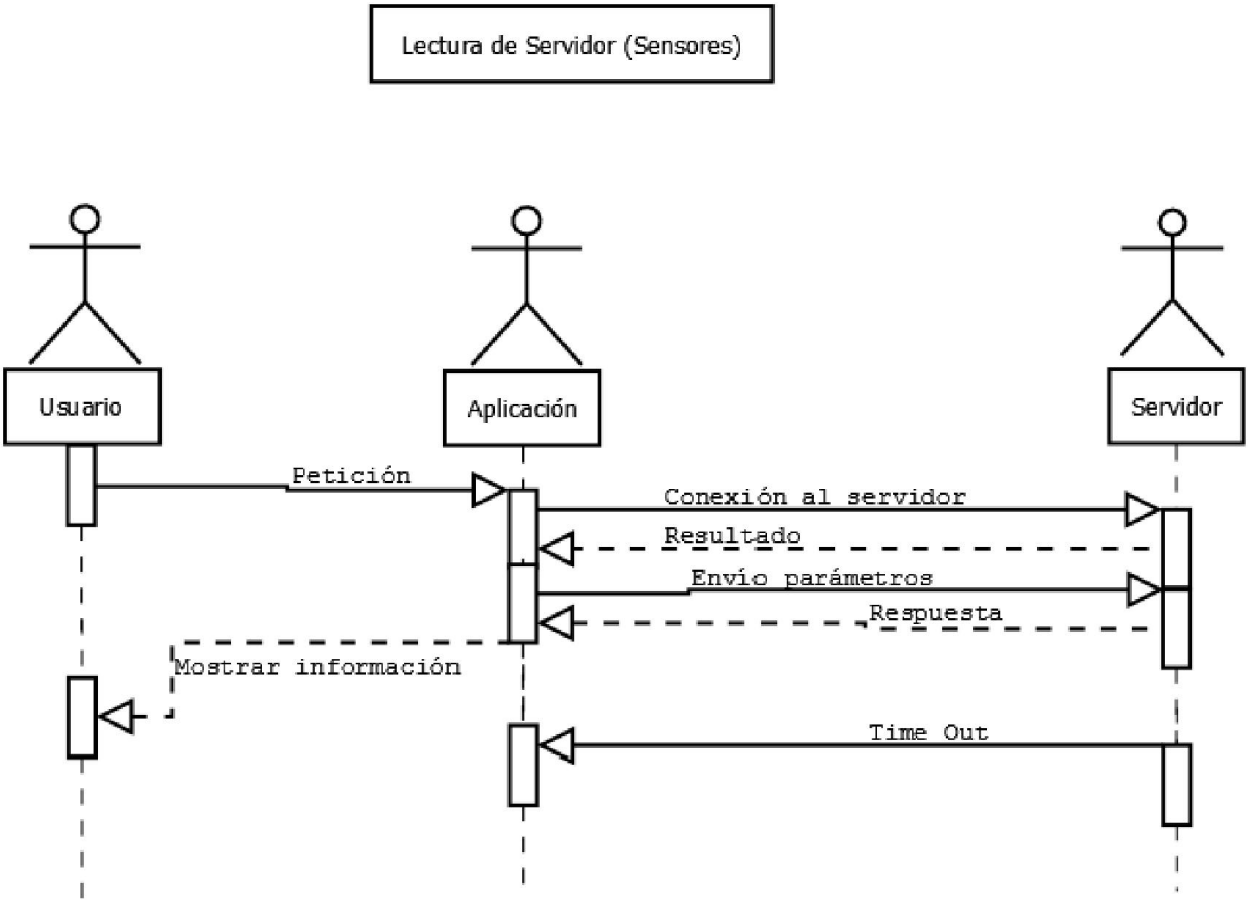

Ilustración 15. D. de Secuencia 1: Monitorización de sensores

La primera y más importante función que la aplicación debe ser capaz de realizar es la obtención de la información de los sensores de una maceta. La información que se obtenga se guarda en un ArrayList para su disponibilidad en caso de no poder establecer conexiones posteriores con el servidor. En esta función intervienen el usuario, la aplicación y el servidor. Es la aplicación la encargada de hacer la petición al servidor, pero en alguna de las peticiones es el usuario el que la desencadena. Para ello, se establece la conexión con el servidor y una vez se recibe el resultado se envían los parámetros necesarios para que el servidor pueda responder con la información correcta. Esta misma estructura es la diseñada para la obtención de las macetas, la información de los sensores de una maceta y el historial de un determinado sensor.

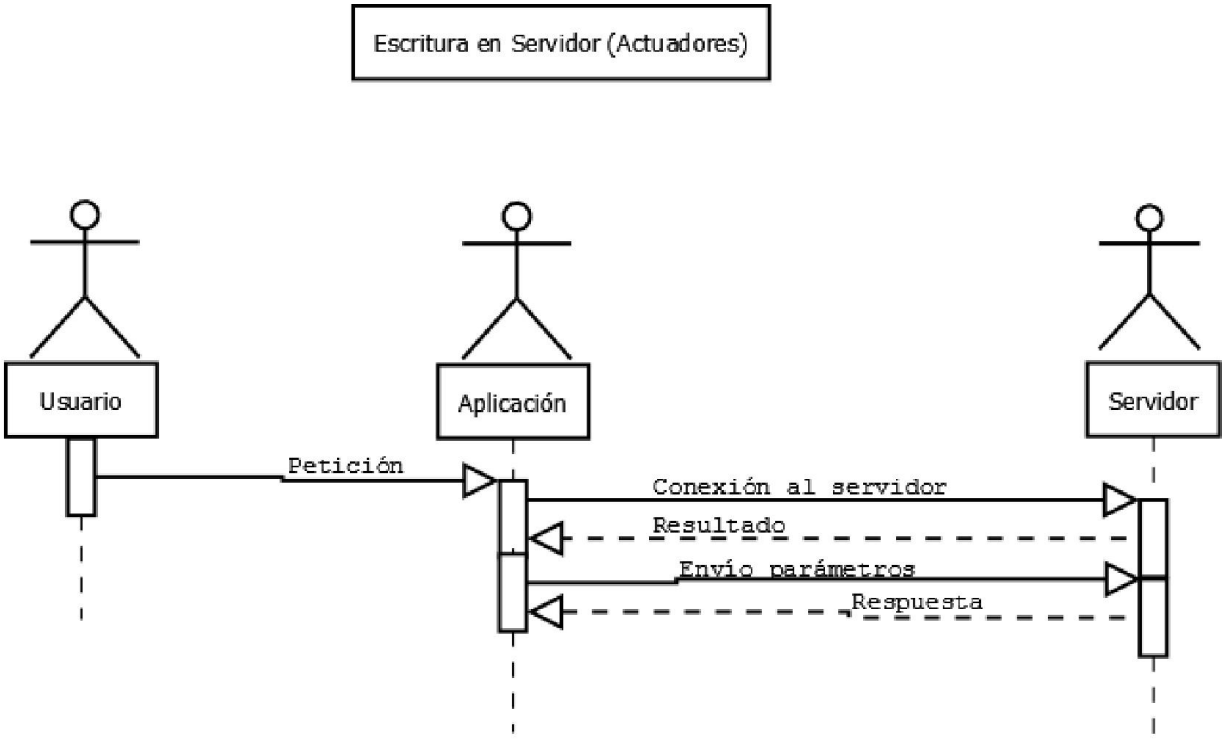

Ilustración 16. D. de Secuencia 2: Actualizar sensores

La segunda función que la aplicación debe ser capaz de realizar es proporcionar al usuario control sobre los actuadores de una maceta. En esta función también intervienen el usuario, la aplicación y el servidor. Cuando el usuario decide accionar un actuador, la aplicación se conecta con el servidor. Si el servidor responde, se le envían los parámetros necesarios para completar la comunicación. Finalmente el servidor responderá y acabará la conexión. En el caso concreto de este proyecto esta secuencia de comunicación se aplicará al control de riego, a la detención de movimiento y al control de movimiento de una maceta.

Después de identificar las secuencias que corresponden a cada caso de uso, se detallan todas las funciones que derivan de los diagramas de secuencia descritos anteriormente:

Código	CFunc_01	Nombre	Monitorización de Sensores
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/sensores/{id_maceta}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. En la misma URL envía los parámetros necesarios, en este caso el id de la maceta. Si se establece la conexión correctamente, el servidor devuelve la respuesta: los datos de los sensores.			
Output (Valores de ejemplo)			
{temperatura_aire:29} {temperatura_suelo:20} {humedad_suelo:20} {luz_1:60} {luz_2:75} {Ultrasonido_1:36} {Ultrasonido_2:25} {NivelAguaSup:100} {NivelAguaInf:0}			

Tabla 14. Función 01: Monitorización de sensores

Código	CFunc_02	Nombre	Actualización de Sensores
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/sensores/{id_maceta}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. En la misma URL envía los parámetros necesarios, en este caso el id de la maceta. Si se establece la conexión correctamente, el servidor devuelve la respuesta: los datos de los sensores.			
Output (Valores de ejemplo)			
{temperatura_aire:29} {temperatura_suelo:20} {humedad_suelo:20} {luz_1:60} {luz_2:75} {Ultrasonido_1:36} {Ultrasonido_2:25} {NivelAguaSup:100} {NivelAguaInf:0}			

Tabla 15. Función 02: Actualización de sensores

Código	CFunc_03	Nombre	Monitorización de historial
Input			
http://IP_SERV:PUERTO/rest/peticion/historial/{id_robot}/sensor/{id_sensor}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. En la misma URL envía los parámetros necesarios, en este caso el id de la maceta y el id del sensor que se desea consultar. Si se establece la conexión correctamente, el servidor devuelve la respuesta: una serie de valores junto con la fecha y hora de su recogida.			
Output (Valores de ejemplo)			
{"2016-05-11 20:19:39.0":0,"2016-05-13 03:23:48.0":100}			

Tabla 16. Función 03: Monitorización del historial

Código	CFunc_04	Nombre	Control de Riego
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_actuador} /{valor_actuador}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. En la misma URL envía los parámetros necesarios, en este caso el id de la maceta, el id del actuador y el valor a enviar. El id de riego es el 2(es un valor fijo) Si se establece la conexión correctamente, el servidor devuelve la respuesta: los datos de los sensores.			
Output			
-			

Tabla 17. Función 04: Control de riego

Código	CFunc_05	Nombre	Detención de movimiento
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_x} http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_y}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. Para esta función se precisan dos URLs, por lo que hay que hacer dos conexiones iguales excepto en el id de sensor que se envía en la URL. El valor X se refiere a izquierda y derecha, y el valor Y se refiere a hacia delante o hacia atrás. Para la detención debe enviarse el valor 0 por cada uno de estos dos parámetros. Si se establece la conexión correctamente, el servidor no devuelve respuesta pero se encarga de procesar la información.			
Output			
-			

Tabla 18. Función 05: Detención de movimiento

Código	CFunc_06	Nombre	Control de Movimiento 1
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_x} http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_y}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. Para esta función se precisan dos URLs, por lo que hay que hacer dos conexiones iguales excepto en el id de sensor que se envía en la URL. El valor X se refiere a izquierda y derecha, y el valor Y se refiere a hacia delante o hacia atrás. Para avanzar debe enviarse el valor 0 para la X y el valor 100 para la Y. Si se establece la conexión correctamente, el servidor no devuelve respuesta pero se encarga de procesar la información.			
Output (Valores de ejemplo)			
-			

Tabla 19. Función 06: Control de movimiento 1

Código	CFunc_07	Nombre	Control de Movimiento 2
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_x} http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_y}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. Para esta función se precisan dos URLs, por lo que hay que hacer dos conexiones iguales excepto en el id de sensor que se envía en la URL. El valor X se refiere a izquierda y derecha, y el valor Y se refiere a hacia delante o hacia atrás. Para retroceder debe enviarse el valor 0 para la X y el valor -100 para la Y. Si se establece la conexión correctamente, el servidor no devuelve respuesta pero se encarga de procesar la información.			
Output (Valores de ejemplo)			
-			

Tabla 20. Función 07: Control de movimiento 2

Código	CFunc_08	Nombre	Control de Movimiento 3
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_x} http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_y}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. Para esta función se precisan dos URLs, por lo que hay que hacer dos conexiones iguales excepto en el id de sensor que se envía en la URL. El valor X se refiere a izquierda y derecha, y el valor Y se refiere a hacia delante o hacia atrás. Para girar hacia la derecha debe enviarse el valor 100 para la X y el valor 0 para la Y. Si se establece la conexión correctamente, el servidor no devuelve respuesta pero se encarga de procesar la información.			
Output (Valores de ejemplo)			
-			

Tabla 21. Función 08: Control de movimiento 3

Código	CFunc_09	Nombre	Control de Movimiento 4
Input			
http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_x} http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_sensor}/{valor_y}			
Procesamiento			
Llamada a tarea asíncrona. Ésta conecta con el servidor vía URL. Para esta función se precisan dos URLs, por lo que hay que hacer dos conexiones iguales excepto en el id de sensor que se envía en la URL. El valor X se refiere a izquierda y derecha, y el valor Y se refiere a hacia delante o hacia atrás. Para girar a la izquierda debe enviarse el valor -100 para la X y el valor 0 para la Y. Si se establece la conexión correctamente, el servidor no devuelve respuesta pero se encarga de procesar la información.			
Output (Valores de ejemplo)			
-			

Tabla 22. Función 09: Control de movimiento 4

Por último en esta sección se mostrarán las clases de las que está compuesto el proyecto y su interacción entre ellas. Dado que la aplicación se realiza con un lenguaje orientado a objetos (Java), cada una de las clases serán las que se procederán a detallar más adelante.

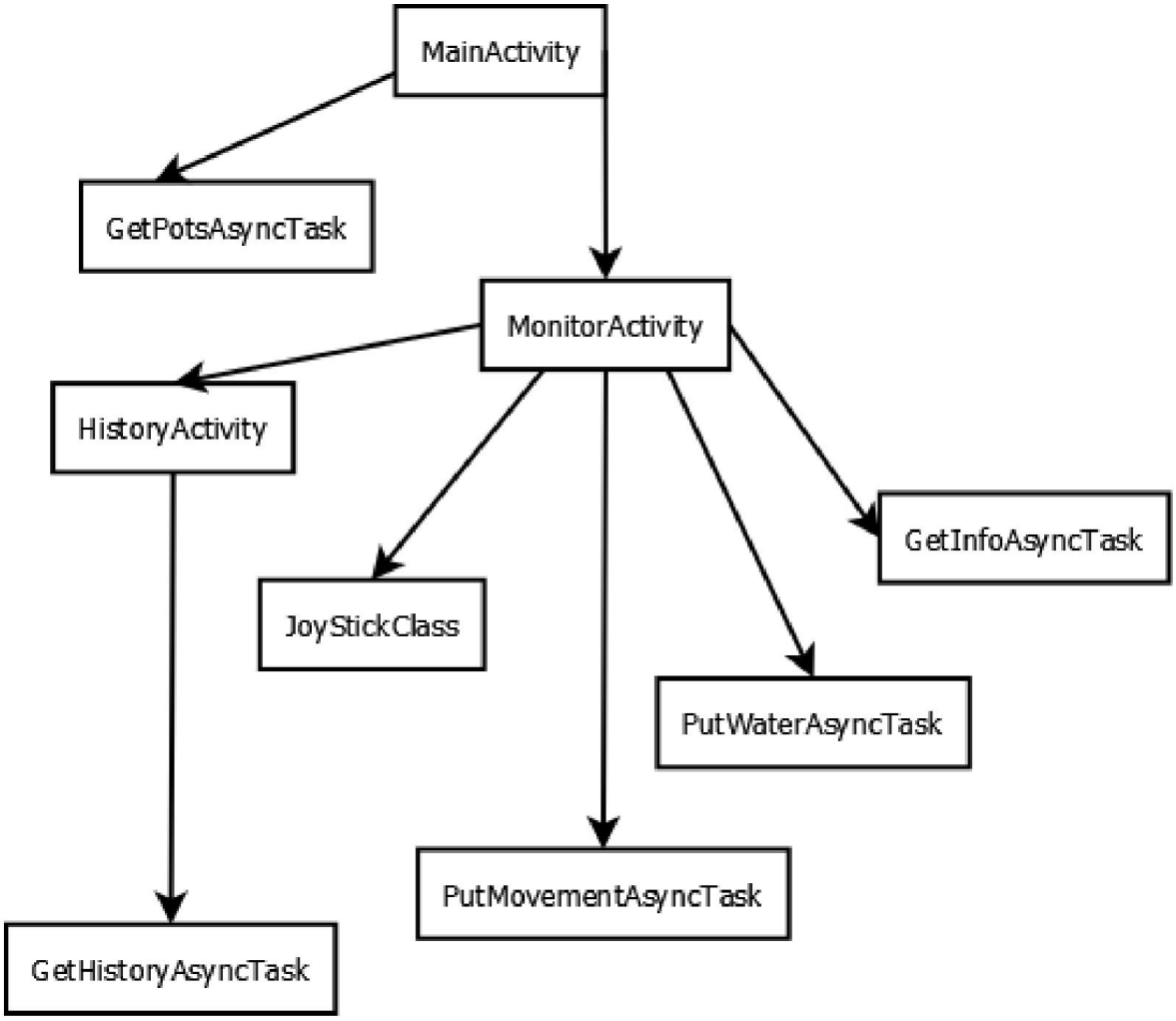

Ilustración 17. Diagrama de bloques

4.3.3 Capa presentación

En esta capa se incluyen los diseños preliminares de la aplicación. Es importante hacer un buen diseño que cumpla ciertas condiciones. La aplicación debe ser fácil de utilizar para el usuario. También debe mostrar toda la información necesaria para el usuario. Por ello, se debe diseñar una aplicación con una estructura sencilla y ordenada.

A continuación se presentan los diseños de cada una de las interfaces que se puede encontrar un usuario en la aplicación, y qué función cumple cada uno de sus elementos.

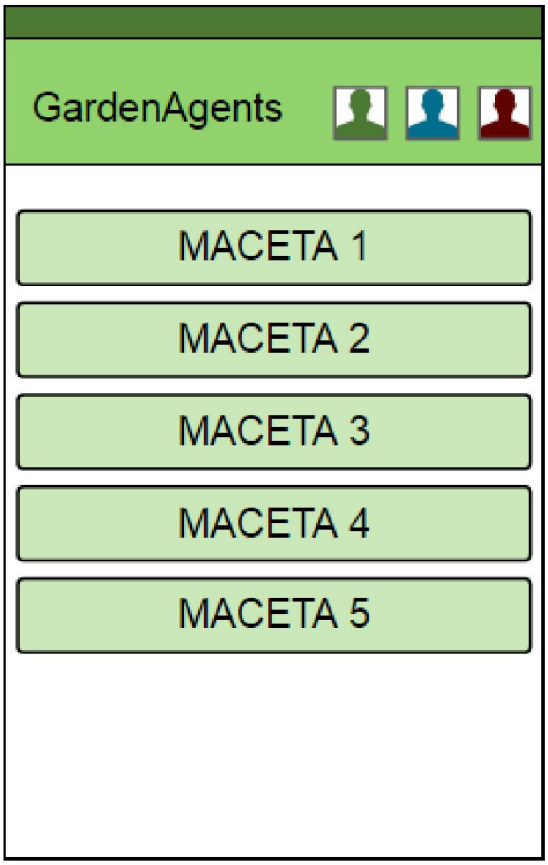

Ilustración 18. Mockup 1

En la ilustración superior se muestra un listado de macetas. Una vez se pulse en una de ellas, se generará una nueva ventana con información referente a la misma. En la parte superior aparecen los tipos de usuario según los cuales se dará acceso a ciertos tipos de permiso, ya sea de consulta o actuación(esto se implementará en un futuro seguimiento del proyecto).

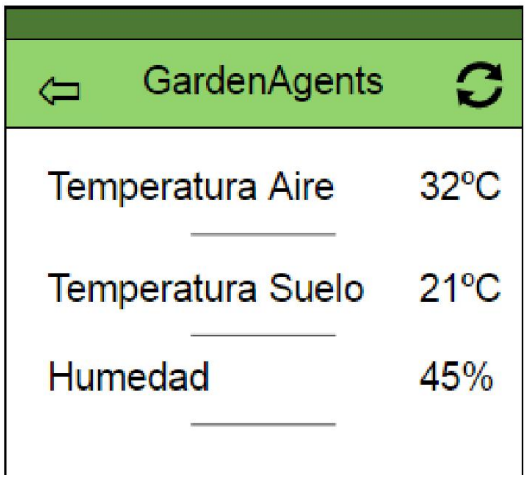

Ilustración 19. Mockup 2

En la ilustración anterior aparecen los sensores con sus respectivos valores. En la parte superior aparece un icono que sirve para la actualización de dichos valores.

Además si se pulsa en alguno de estos sensores aparecerá otra ventana con el historial de dicho sensor, que incluye la fecha y la hora de la toma de valor y el propio valor. El diseño de esta nueva ventana se muestra en la siguiente ilustración.

Todos estos datos se muestran directamente a partir de la información que el servidor proporciona a través de su base de datos.

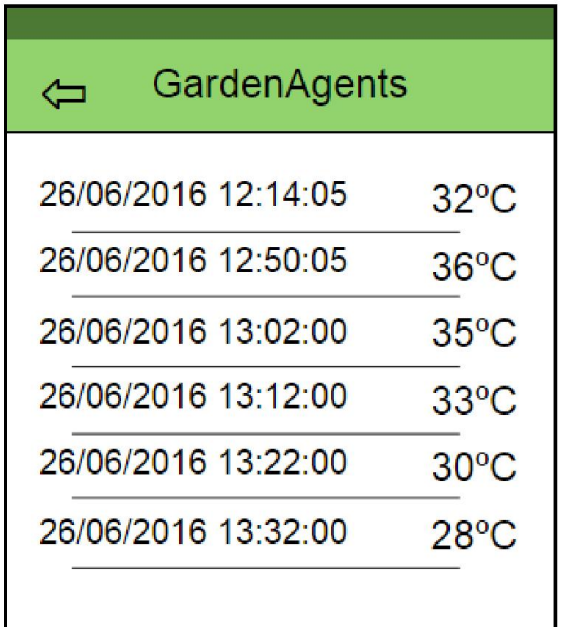

Ilustración 20. Mockup 3

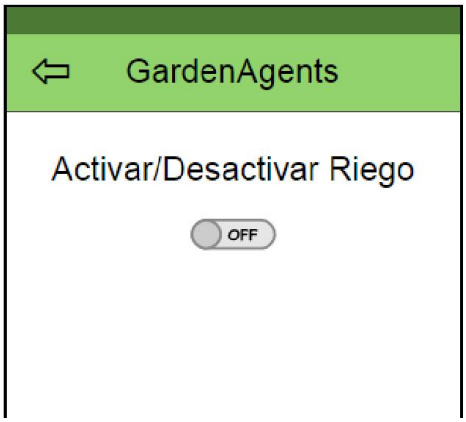

Ilustración 21. Mockup4

En la ilustración superior se puede ver que se va a utilizar un interruptor para activar y desactivar el riego. Es un método simple y muy intuitivo para el usuario.

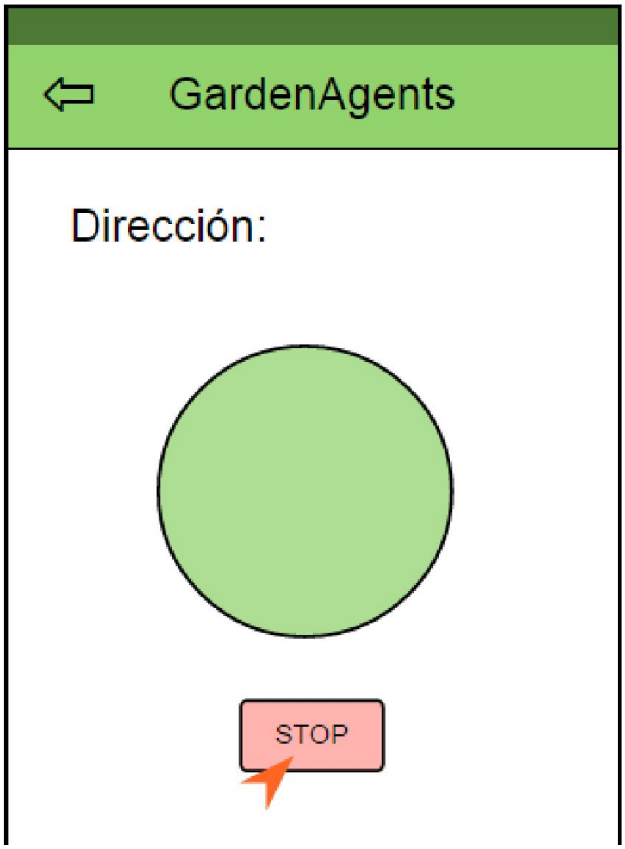

Ilustración 22. Mockup5

En la ilustración superior se puede ver cómo será la distribución respecto al control de movimiento. Puede verse el texto "Dirección:" que mostrará la dirección mientras se haga uso del joystick. El joystick en sí se utilizará desplazando el dedo por encima del círculo hacia la dirección deseada.

Además hay un botón debajo con la palabra stop, que se utiliza para detener el movimiento de la maceta.

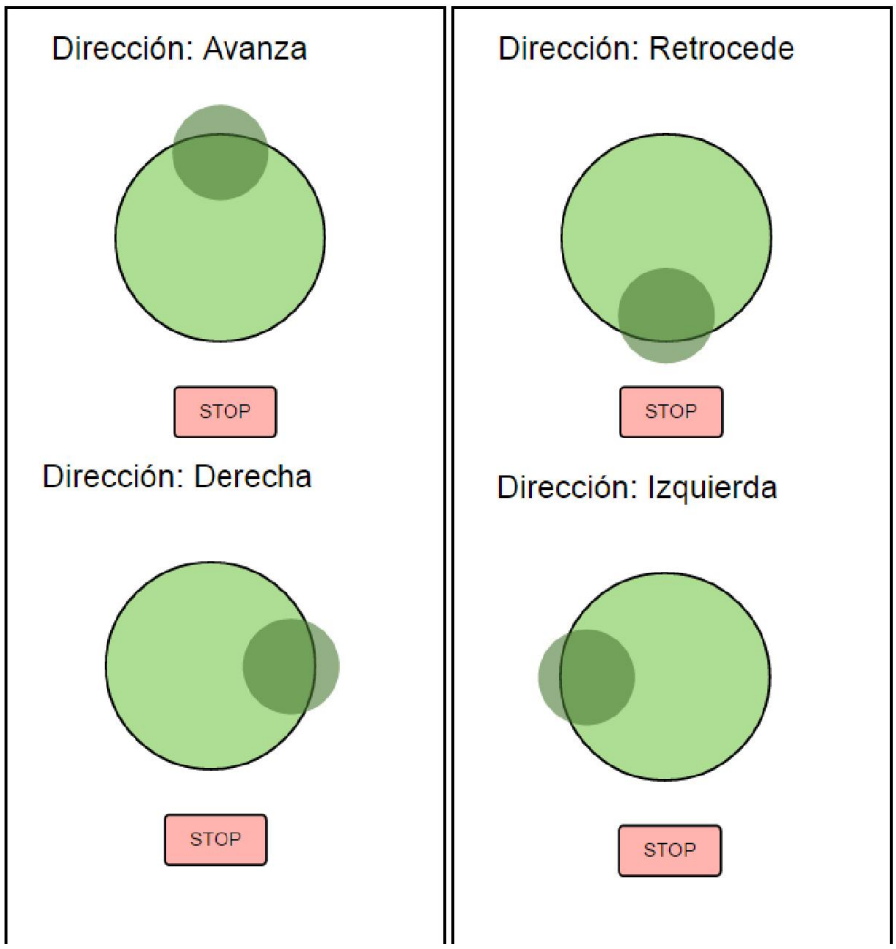

Ilustración 23. Mockup 6

En la ilustración superior se muestra cómo será la apariencia del joystick durante el su uso. Se aprecia que, junto con la dirección en texto que se muestra en pantalla, aparece un círculo pequeño que representa la posición exacta de la pantalla que el usuario está pulsando.

4.4 Conclusiones

En esta sección se ha realizado una especificación formal para demostrar que es necesario añadir un control inteligente para tener en todo momento la maceta o macetas totalmente bajo control en caso de un mal funcionamiento por parte de alguno de los dispositivos involucrados en el sistema. Se ha optado por una arquitectura de 3 capas que cubre las necesidades de esta parte del sistema.

Además, se ha diseñado mediante tareas asíncronas para evitar el colapso del hilo principal que se encarga de controlar la interfaz de un dispositivo móvil. También se realizan las comprobaciones necesarias para poder obtener datos sin provocar el cierre inesperado de la aplicación.

Por último, se ha diseñado el aspecto general que debe tener la aplicación. Se aspira a conseguir una aplicación con una interfaz sencilla para el usuario pero completa a la vez y que ofrezca todas las funcionalidades requeridas. Es un diseño intuitivo y funcional. Con este diseño se ha conseguido evitar una saturación de información y al mismo tiempo se mantiene un orden en la distribución de la información y transmitir al usuario una sensación de comodidad.

5 Implementación, implantación y evaluación

5.1 Introducción

En esta sección se va a detallar la forma en que se han implementado las funcionalidades descritas en la sección anterior. Para ello se describe la estructura que se ha seguido para desarrollar toda la aplicación.

Además se describe el proceso de implementación por parte del usuario y luego se describen las evaluaciones que se han realizado para asegurar el correcto funcionamiento de todos los casos de uso.

5.2 Implementación

En esta sección se van a exponer los detalles de la implementación. Se empieza por la capa de presentación porque en este caso es la capa más importante, puesto que es la que está en contacto directo con el usuario y debe realizarse minuciosamente. Además, es más sencillo implementar las demás capas una vez se tiene el diseño de la capa de presentación que empezar por la capa de persistencia.

5.2.1 Capa de presentación

Para describir la capa de presentación primero debemos describir qué actividades compone la aplicación y qué diseño tiene cada una de ellas.

- MainActivity: Es la actividad donde se muestran todas las macetas. Tiene un formato de lista, de forma que los elementos quedan ordenados y accesibles fácilmente. Cada maceta está representada por un botón. Mediante la pulsación de cada uno de ellos, se desencadenarán una serie de acciones. También aparecen 3 iconos para una posible mejora del proyecto en un futuro.

- MonitorActivity: Esta actividad se compone de tres pestañas diferenciadas: la pestaña de sensores, la de actuadores y la de motor. Además aparece un icono en la barra superior para la actualización de los valores de los sensores.

En la pestaña de sensores hay una distribución lineal, al igual que la actividad principal. Se ha optado por este tipo de distribución porque puede mostrarse mucha información al mismo tiempo pero de forma ordenada para que sea clara para el usuario.

La siguiente pestaña, la de actuadores, por ahora sólo muestra un interruptor de encendido/apagado para controlar el riego. Esto se debe a que el diseño final de la maceta no tiene más actuadores, pero si en un futuro se añaden

más, en esta sección pueden añadirse sus correspondientes elementos de control.

La tercera pestaña es la del motor. En esta pestaña se ve un círculo, seguido de un botón de STOP. La utilización del *joystick*(que se detallará su construcción a continuación) es muy sencilla: tan sólo hay que desplazar el dedo por la pantalla dentro de este círculo, y se mostrará por pantalla la dirección que se enviará a la maceta como orden. Por otro lado, el botón sirve para detener la maceta en cuestión.

-HistoryActivity: Es la actividad donde se muestra el historial completo de valores de un sensor. Tiene formato de lista y muestra la fecha y la hora exactas en que se toma la medición, y el valor recogido.

Por último en esta capa se presenta la clase JoyStickClass, una clase construida para la implementación del joystick. En esta clase se utilizan dos imágenes: la primera es un círculo; una imagen estática que actúa como zona de interacción. Es decir, si se arrastra el dedo por una zona de la pantalla alejada de dicho círculo el joystick no se activará. Sin embargo, si se arrastra dentro del círculo aparece la segunda imagen justo donde se toca la pantalla.

Para mostrar todo esto hay que realizar una serie de cálculos:

- Calcular el ángulo a partir de la posición X e Y:El ángulo lo vamos a necesitar para mostrar la dirección correcta y para mostrar el círculo en la posición correcta.

```
private double cal_angle(float x, float y) {
    if(x >= 0 && y >= 0)
        return Math.toDegrees(Math.atan(y / x));
    else if(x < 0 && y >= 0)
        return Math.toDegrees(Math.atan(y / x)) + 180;
    else if(x < 0 && y < 0)
        return Math.toDegrees(Math.atan(y / x)) + 180;
    else if(x >= 0 && y < 0)
        return Math.toDegrees(Math.atan(y / x)) + 360;
    return 0;
}
```

Ilustración 24. Cálculo de ángulo

- Obtener la dirección que se ha indicado: Este valor se calcula a partir del ángulo calculado anteriormente. Inicialmente se implementó para abarcar también direcciones diagonales, pero éstas se han descartado debido a la complejidad a la hora de comunicarlas con el servidor y la maceta.

```
public int get8Direction() {
    if(distance > min_distance && touch_state) {
        if(angle >= 247.5 && angle < 292.5 ) {
            return STICK_UP;
        } else if(angle >= 292.5 && angle < 337.5 ) {
            return STICK_UPRIGHT;
        } else if(angle >= 337.5 || angle < 22.5 ) {
            return STICK_RIGHT;
        } else if(angle >= 22.5 && angle < 67.5 ) {
            return STICK_DOWNRIGHT;
        } else if(angle >= 67.5 && angle < 112.5 ) {
            return STICK_DOWN;
        } else if(angle >= 112.5 && angle < 157.5 ) {
            return STICK_DOWNLEFT;
        } else if(angle >= 157.5 && angle < 202.5 ) {
            return STICK_LEFT;
        } else if(angle >= 202.5 && angle < 247.5 ) {
            return STICK_UPLEFT;
        }
    } else if(distance <= min_distance && touch_state) {
        return STICK_NONE;
    }
    return 0;
}
```

Ilustración 25. Obtener dirección

Una vez realizados estos cálculos, se implementa la función que dibuja el círculo que simula el movimiento del dedo. Este método incluye una variable de compensación(OFFSET) para que el movimiento no sea tan brusco. Las direcciones de avanzar y retroceder, que se indican al mover el círculo arriba y abajo es más fácil de representar, pues sólo hay que tener en cuenta el ancho de la pantalla. Sin embargo, para las direcciones de izquierda y derecha (que en un principio incluían direcciones diagonales) hay que hacer los cálculos que se muestran en la ilustración para obtener la posición exacta.

```
public void drawStick(MotionEvent arg1) {
    position_x = (int) (arg1.getX() - (params.width / 2));
    position_y = (int) (arg1.getY() - (params.height / 2));
    distance = (float) Math.sqrt(Math.pow(position_x, 2) + Math.pow(position_y, 2));
    angle = (float) cal_angle(position_x, position_y);

    if(arg1.getAction() == MotionEvent.ACTION_DOWN) {
        if(distance <= (params.width / 2) - OFFSET) {
            draw.position(arg1.getX(), arg1.getY());
            draw();
            touch_state = true;
        }
    } else if(arg1.getAction() == MotionEvent.ACTION_MOVE && touch_state) {
        if(distance <= (params.width / 2) - OFFSET) {
            draw.position(arg1.getX(), arg1.getY());
            draw();
        } else if(distance > (params.width / 2) - OFFSET){
            double aux_X = (Math.cos(Math.toRadians(cal_angle(position_x, position_y))));
            float x = (float) aux_X * ((params.width / 2) - OFFSET);
            double aux_Y = (Math.sin(Math.toRadians(cal_angle(position_x, position_y))));
            float y = (float) aux_Y * ((params.height / 2) - OFFSET);
            x += (params.width / 2);
            y += (params.height / 2);
            draw.position(x, y);
            draw();
        } else {
            mLayout.removeView(draw);
        }
    } else if(arg1.getAction() == MotionEvent.ACTION_UP) {
        mLayout.removeView(draw);
        touch_state = false;
    }
```

Ilustración 26. Dibujo de círculo pequeño

Por último, se puede ver que se ejecuta un método draw() dentro de cada dirección. Este método es el encargado de eliminar el círculo de su posición antigua y redibujarlo.

```
private void draw() {
    try {
        mLayout.removeView(draw);
    } catch (Exception e) { }
    mLayout.addView(draw);
}
```

Ilustración 27. Método de dibujo

5.2.2 Capa de negocio

En esta capa se describen todas las tareas asíncronas que se utilizan en la aplicación. Las tareas asíncronas son hilos adicionales que se crean y ejecutan instrucciones en segundo plano para no entorpecer el hilo principal, que es el encargado de gestionar la interfaz de usuario. Por tanto todas las operaciones de comunicación con el servidor, que suelen ser costosas, se realizan a través de estas tareas. Para implementar una serie de acciones en una tarea asíncrona se debe construir el método doInBackground.

En este proyecto podemos encontrar cinco. Las primeras dos tareas utilizan un pequeño método de conversión a string que se comentará más adelante:

- GetInfoAsyncTask: Esta tarea se encarga de obtener la información de los sensores de una maceta. Cada entrada de sensor se guarda en un HashMap con 3 entradas o claves: descripción, valor e icono. El valor de descripción es el nombre del sensor que se mostrará al usuario. La información que guarda valor es el valor a mostrar (añadiendo la unidad correspondiente). Por último se guarda icono, una referencia al icono correspondiente a mostrar junto al sensor para identificarlo más rápidamente.

- GetHistoryAsyncTask: Esta tarea se encarga de obtener todas las mediciones de un sensor de una planta. Cada entrada se guarda en un HashMap al igual que la tarea anterior. Sin embargo en éste sólo están las claves de descripción y valor, en las que se guarda la fecha de la obtención y el valor.

- PutWaterAsyncTask: Esta tarea es la encargada de enviar la orden de activación o desactivación de la bomba de agua. Dado que los parámetros se envían en la misma URL con la que se establece la conexión, no hace falta más que establecer la conexión.

- PutMovementAsyncTask: Esta tarea se encarga de enviar la dirección que debe tomar la maceta, o la orden de detención. Dado que los parámetros se envían en la misma URL con la que se establece la conexión, no hace falta más que establecer la conexión con el servidor mediante la misma.

- GetPotsAsyncTask: Por último esta tarea es la encargada de recibir el número de macetas que existen, para así mostrar la misma cantidad de botones en la actividad principal. Finalmente esta tarea no se llama desde MainActivity dado que sólo se tiene una maceta funcional en la base de datos. Se muestran pues 5 botones de ejemplo, pero sólo el primero(el correspondiente a la maceta 1) tiene información válida en el servidor.

Para finalizar la descripción de esta capa, se pasa a detallar que código contiene la actividad MonitorActivity para enlazar toda la información recibida con la interfaz diseñada:

- onPostExecute: Este método, incluido en la tarea asíncrona GetInfoAsyncTask, ejecuta el método que se muestra en la siguiente imagen. Éste se ejecuta al acabar la comunicación con el servidor, y contiene el adapter para enlazar la información con la interfaz:

```
public void setListPos(ArrayList<HashMap<String, String>> result){
    hashMapList.addAll(result);
    if(hashMapList!=null) {
        //Adapter para enlazar la información con la interfaz
        adapter1 = new SimpleAdapter(this, hashMapList, R.layout.monitor_list_row,
                new String[]{"icono", "descripcion", "valor"},
                new int[]{R.id.icon_monitor, R.id.tvtipo, R.id.tvvalor});
        sensorsListView.setAdapter(adapter1);
    }
}
```

Ilustración 28. Método de enlace a interfaz

Para el diseño de lista de la actividad de monitorización se ha creado un recurso en el que se describe cómo debe estar distribuida la información en cada línea de la lista:

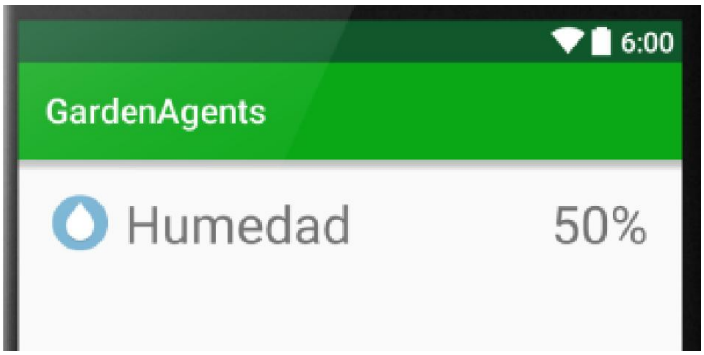

Ilustración 29. Diseño de línea

Finalmente, tanto para el botón de STOP como para el interruptor, hay un método que "escucha" el evento de pulsación y actúa en consecuencia cuando éste se detecta. En este caso, se lanza a ejecución la tarea asíncrona correspondiente.

5.2.3 Capa de persistencia

Esta capa está orientada al guardado de la información. En este caso, la mayoría de la información la contiene el servidor y se recupera de ahí. Sin embargo, hay un elemento importante en la aplicación que nos garantiza recoger la información correcta en cada momento: las SharedPreferences.

Las SharedPreferences son unas configuraciones compartidas por la aplicación entera, por lo que no dependen de la clase en la que se creen. Son

accesibles desde todas las clases, y eso es una gran ventaja a la hora de consultar datos desde diferentes sitios y en distintos momentos.

Las SharedPreferences tienen una estructura en forma de clave y valor, por lo que a cada valor o dato almacenado debe asociársele una clave. Por tanto es importante documentar las claves utilizadas, asignar unas etiquetas de clave que indiquen de forma clara que información contiene. Las etiquetas utilizadas en este proyecto son:

- id_maceta: guarda en forma de entero el id de la maceta que se está monitorizando.

- id_sensor: guarde en forma de entero el id del sensor que se está monitorizando.

-bomba_agua: guarda en forma de entero el valor que se debe enviar al servidor (100 para activarlo y 10 para desactivarlo).

La forma de guardar la información en esta estructura es llamar al PreferenceManager, que se encarga de coger estas preferencias. A continuación, se utiliza un editor para cambiar la información que contiene. Se introducen todos los cambios deseados, y una vez hecho esto se deben aplicar. Si no se aplican los cambios no tendrá efecto todo cambio realizado con el editor. En la siguiente ilustración se muestra un ejemplo de los pasos a seguir:

```
prefs = PreferenceManager.getDefaultSharedPreferences(this);
editor = prefs.edit();
editor.putInt("id_maceta",1);
editor.apply();
```

Ilustración 30. Manipulación de SharedPreferences

5.3 Implantación

Una vez completada la aplicación tan sólo falta su implantación para que los usuarios puedan hacer uso de ella. Para ello, tan sólo deben seguirse una serie de pasos y estará lista para su utilización.

Primero debe obtenerse un archivo con extensión 'apk' para que pueda ser instalada en un dispositivo. Para ello, desde el propio programa de desarrollo Android Studio se debe acceder al menú Build, y dentro de éste seleccionar la opción Build APK. Esto generará dicho archivo en le carpeta GardenAgents\app\build\outputs\apk

A continuación se debe transferir este archivo al dispositivo en el cual se quiera instalar. Debe tenerse en cuenta que la aplicación únicamente podrá instalarse en dispositivos (tanto móviles como tablets) con una versión Android

igual o superior a la 4.0.3. Esto se debe a que implementar una aplicación que sea compatible con versiones anteriores limita bastante las herramientas disponibles para su implementación. El archivo puede transferirse mediante conexión USB al ordenador que lo contenga, puede descargarse de internet si se aloja el archivo en la nube, etc.

Una vez se ha transferido el archivo, debe buscarse mediante una aplicación de explorador de archivos. Por defecto el dispositivo lleva uno, pero si éste no fuera el caso hay aplicaciones gratuitas que ofrecen esta funcionalidad en el Google Play Store. Hay que comprobar que previamente se ha activado la opción orígenes desconocidos en el apartado de Seguridad de los Ajustes de dispositivo. Si no se activa esta opción no será posible la instalación.

Por último tan sólo debe ejecutarse el archivo y empezará su instalación. Una vez terminada estará lista para ser utilizada por los usuarios.

5.4 Evaluación

Una vez la implementación está terminada, e incluso durante la misma, se realizan todas las evaluaciones necesarias sobre la aplicación para asegurarse de que ésta funciona correctamente.

Prueba	Cobertura	Resultado
Test01	CU_01 → CFunc_01	Si se selecciona una maceta muestra los sensores que tiene ésta y sus valores actuales.
Test 02	CU_02→ CFunc_02	Si se pulsa el botón de actualización se recuperan los valores más actuales de los sensores de una maceta.
Test 03	CU_03 → CFunc_03	Si se pulsa sobre alguno de los sensores se puede ver todo el historial de éste.
Test 04	CU_04 → CFunc_04	Si se pulsa el activador del riego el servidor recibe dicha orden.
Test 05	CU_05 → CFunc_05	Si se pulsa el botón STOP el servidor recibe la orden correctamente.
Test 06	CU_06 → CFunc_06, CFunc_07, CFunc_08, CFunc_09	Al efectuar movimientos con el joystick en las cuatro direcciones (arriba, abajo, izquierda y derecha), el servidor recibe correctamente cada una de las órdenes.

Tabla 23. Pruebas de evaluación

6 Conclusiones

Por último esta sección está enfocada a mostrar qué dificultades se han encontrado a lo largo del desarrollo del proyecto, y qué decisiones han llevado a resolverlas. También se hace una pequeña introducción a posibles mejoras o añadidos que puedan aparecen en años posteriores, puesto que otros alumnos de la universidad han mostrado interés en mejorar y/o ampliar cada una de las partes que compone el sistema.

6.1 Aportaciones

En primer lugar se han analizado las aplicaciones existentes en el mercado, en concreto en el campo de la automatización de la jardinería. Para ello se ha hecho la correspondiente tabla comparativa con diversas características para posteriormente extraer las características que se deben encontrar en este proyecto. Esta sección del proyecto puede servir en un futuro como punto de partida de otros proyectos tanto del campo de la jardinería como del desarrollo de aplicaciones para plataforma Android.

A continuación, a partir de las características extraídas, se ha decidido que estructura tendrán las comunicaciones y qué diseño tendrá la aplicación, cuyo objetivo es que la aplicación tenga un diseño funcional, intuitivo y simple con respecto a las aplicaciones similares encontradas en el mercado.

A partir de ello, se ha hecho un diseño con arquitectura de tres capas: persistencia, negocio y presentación. Ésta permite diferenciar los elementos que conforman el proyecto

Por último se ha descrito el procedimiento de implementación e implantación de la aplicación junto con la descripción de los problemas encontrados durante su realización y algunas futuras ampliaciones interesantes.

6.2 Dificultades resueltas

Se van a exponer algunas dificultades que han surgido durante la realización del proyecto.

El hecho de realizar un proyecto que forma parte de un sistema de 4 personas implica comunicación y cooperación. En este caso otros miembros debían definir aspectos de sus proyectos para tener los datos necesarios para avanzar ciertos aspectos de éste. En concreto, hasta que no se sabía con certeza qué sensores y actuadores habría en la maceta, no se sabía que información habría en la base de datos. Y hasta que no se sabía el contenido de la base de datos, no se podían construir las URLs del servidor REST. Sin todos estos datos, se estuvieron barajando distintos diseños con diferentes posibilidades y funcionalidades. Finalmente, ya con los actuadores disponibles y las URLs definidas, se ha terminado el proyecto.

Otro de los problemas que se han hallado durante el proceso es adaptar el formato que tiene la información al ser recibida del servidor. Puesto que la persona encargada del servidor formaba parte del equipo, se ha podido modificar en parte dicho formato para facilitar el proceso desde la aplicación. Además, para simplificar el tratamiento de los datos en formato JSON[5]y así no depender de librerías externas para el proceso de este tipo de objetos, se ha añadido un método de tratamiento de respuesta que convierte a String la información, para después tratarla de manera más sencilla:

```
public static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
    return s.hasNext() ? s.next() : "";
}
```

Ilustración 31. Método de conversión a string

6.3 Futuras implementaciones

Después de describir la totalidad el proyecto, se presentan unas futuras implementaciones que pueden llevarse a cabo por otros alumnos interesados en la continuidad de éste.

Una de las posibilidades reside en la implementación de diferentes permisos dependiendo del tipo de usuario que va a tener acceso a la aplicación. Esto puede realizarse aprovechando la infraestructura ya creada para este propósito.

Por otra parte otro elemento de interés puede ser la restricción de acceso mediante un sistema de inicio de sesión con usuario y contraseña. Esto permitiría que la aplicación estuviera al alcance de cualquier persona, pero sólo funcional para determinadas personas.

Si en un futuro la maceta tuviera una mayor cantidad de actuadores, también sería interesante implementar algún tipo de interacción mediante la aplicación.

Todas estas propuestas quedan a disposición de los alumnos que estén interesados en ampliar y/o mejorar este proyecto.

7 Referencias

7.1 Bibliográficas

[1]. WEBSTER, John G.; EREN, Halit (ed.). Measurement, Instrumentation, and Sensors Handbook: Spatial, Mechanical, Thermal, and Radiation Measurement. CRC press, 2014.

[2]. JANOCHA, Hartmut (ed.). *Actuators: basics and applications*. Springer Science & Business Media, 2013.

[3]. ZAPATA, Belén Cruz. *Android studio application development*.Packt Publishing Ltd, 2013.

7.2 Internet

1. Android Developers:https://developer.android.com (Acceso: Junio 2016)

2. Stack Overflow: http://es.stackoverflow.com (Acceso: Junio 2016)

3. Contenido Multimedia:https://www.youtube.com/user/DebaterOfMath (Acceso: Mayo 2016)

3. Github:https://github.com/zerokol/JoystickView (Acceso: Junio 2016)

4. Google Play Store: https://play.google.com/store?hl=es (Acceso: Mayo 2016)

5. Apple iTunes: https://itunes.apple.com (Acceso: Mayo 2016)

6. Android for beginners: http://androidforbeginners.blogspot.com.es (Acceso: Abril 2016)

7. Draw.io: https://www.draw.io/ (Acceso: Junio 2016)

8. Moqups: https://moqups.com/ (Acceso: Junio 2016)

9. Pixlr: https://pixlr.com/ (Acceso: Junio 2016)

7. Bitbucket: https://bitbucket.org/ (Acceso: Mayo 2016)