UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Control vía móvil de un jardín inteligente

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alberto Ramírez Losilla

Tutor: José Luis Poza Lujan

Curso académico 2013-2014

Agradecimientos

El presente proyecto no habría sido posible sin la inestimable ayuda y apoyo de una serie de personas que han contribuido de una manera u otra en la realización del mismo.

Primero de todo, agradecer al director del proyecto, el Doctor José Luis Poza Luján, el gran compromiso con el proyecto y la ayuda técnica y didáctica proporcionada.

Por otro lado, quisiera dar la enhorabuena a todos los compañeros encargados de otros proyectos que forman parte de ArduEntorno por el gran trabajo de equipo realizado.

Por último y no menos importante, quiero agradecer todo el apoyo y ánimo recibido por parte de familiares y amigos.

Resumen

El presente proyecto (iEnvironment) está integrado dentro de un proyecto de mayor envergadura (ArduEntorno) enfocado a desarrollar un sistema de control inteligente de entornos exteriores.

El proyecto se centra en el desarrollo de una aplicación para monitorizar y gestionar el sistema de control inteligente desde dispositivos móviles Android, los servicios web con los que interactúa la aplicación y una base de datos para la persistencia del estado del sistema.

Para desarrollar lo anteriormente mencionado, previamente se ha analizado el entorno del proyecto, se han especificado los requisitos funcionales y no funcionales del sistema, se han diseñado aspectos fundamentales como la interfaz de usuario, la base de datos y la arquitectura del sistema y se han tomando decisiones de implementación, como los lenguajes seleccionados, los entornos de desarrollo utilizados y las APIs integradas en el proyecto.

Palabras clave: Arduino, Android, domótica, control de entornos exteriores, servicios web REST, JAX-RS, JPA, MySQL

Tabla de contenidos

Índice de Ilustraciones

Índice de Tablas

Índice de Código

1. Introducción

1.1. Presentación del tema

La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema (Asociación Española de Domótica e Inmótica). El proyecto se centra en espacios exteriores, como pueden ser jardines, espacios lúdicos, plantaciones, etc.

El control del entorno va más allá de una simple automatización de tareas. Se busca un control inteligente del entorno, es decir, que el sistema pueda tomar decisiones en función del estado de dicho entorno.

1.2. Motivación

La domótica ha evolucionado con el paso del tiempo y está cada vez más presente en nuestras vidas, ofreciendo soluciones completas a todo tipo de viviendas. Desde el nacimiento de la domótica, ésta se ha centrado en el interior de la vivienda, siendo el exterior de la misma el gran olvidado.

Además, cualquier sistema de domótica tiene que facilitar la comunicación entre el usuario y el propio sistema. Esto se puede conseguir mediante una aplicación para dispositivos móviles (app), ya que la gran mayoría de las personas disponen de un Smartphone o Tablet que usan frecuentemente. Este tipo de dispositivos permiten utilizar la app desde cualquier lugar (siempre y cuando se disponga de conexión a Internet), ofreciendo la posibilidad de monitorizar y gestionar el estado del sistema.

1.3. Objetivos

El objetivo fundamental que persigue el proyecto es desarrollar una aplicación para dispositivos móviles que permita al usuario comunicarse con ArduEntorno desde cualquier lugar, de forma fácil y segura. Para conseguir esto, la aplicación tendrá que tener una configuración sencilla y una interfaz intuitiva. Por otro lado, toda la información enviada y recibida por la app será cifrada y habrá control de usuarios para asegurar una control de acceso .

1.4. Descripción del documento

La trabajo se organiza de la siguiente forma: El apartado 2 estudia el entorno del proyecto, exponiendo aspectos importantes que hay que tener en cuenta a la hora de desarrollar apps para Android, y analiza aplicaciones similares que están disponibles en el mercado. El apartado 3 detalla la especificación de requisitos siguiendo el estándar IEEE830 . El apartado 4 trata aspectos relativos al diseño conceptual y formal del sistema, como los prototipos de la aplicación, el modelo de la base de datos, la arquitectura del sistema, etc. El apartado 5 expone información relativa a la implementación del sistema, como por ejemplo, los lenguajes seleccionados, las APIs y los entornos de desarrollo utilizados para la elaboración del proyecto y ejemplos de código. Por último, se exponen las conclusiones obtenidas.

2. Entorno

2.1. Introducción

En este apartado se exponen detalles que hay que tener en cuenta para desarrollar aplicaciones para dispositivos móviles con Sistema Operativo Android.

Por otro lado, se hace un análisis cuantitativo y cualitativo de aplicaciones similares disponibles en el mercado.

Estudiando toda esta información, se realiza un análisis estratégico para diferenciar la app dentro del mercado de la domótica y control de exteriores.

Por último, se resume toda la información vista en el apartado con una serie de conclusiones.

2.2. Entorno del proyecto

Un detalle muy importante que hay que tener en cuenta es la gran fragmentación que existe en la versiones del Sistema Operativo utilizado por los usuarios de dispositivos Android.

Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%

Data collected during a 7-day period ending on May 1, 2014.
Any versions with less than 0.1% distribution are not shown.

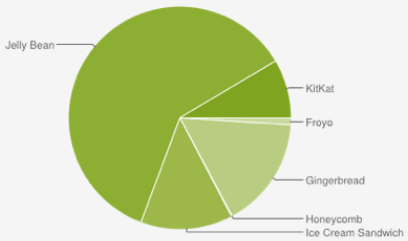

Ilustración 1: Fragmentación de Android **(Android Open Source Project, 2014)**

Por otro lado, también hay que analizar las características que ofrece cada versión y llegar al máximo número de usuarios excluyendo lo menos posible.

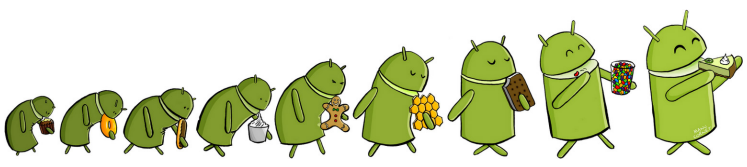

Ilustración 2: Evolución de Android **(Cornet, 2012)**

A continuación se muestra en una tabla las diferentes versiones de Android y las características que ofrece cada una:

Lanzamiento	Nombre Comercial	Versión	API	Características relevantes
23/09/2008	Apple pie	1.0	1	No se comercializó.
09/02/2009	Banana bread	1.1	2	No se comercializó. Reseñas en mapas, guardar adjuntos, etc.
30/04/2009	Cupcake	1.5	3	1ª versión comercial. Teclado en pantalla, widgets.
15/09/2009	Donut	1.6	4	Búsqueda avanzada, soporte a Gesto multi-touch, texto a voz, diferentes densidades de pantalla, atributo de Vista onclick.
26/10/2009	Éclair	2.0/2.1	5-7	Sin usuarios, HTML5, servicio centralizado de cuentas, Bluetooth 2.1, fondos animados, webkit, bases de datos, configuración de la caché. Llegó al 99.9% de los usuarios del momento.
20/05/2010	Froyo	2.2x	8	Compilador JIT (ejecución más rápida), permite instalar apps en memoria externa, notificaciones push, interacción con usuarios por voz, modos "automóvil" y "noche".
06/12/2010	Gingerbread	2.3.x	9-10	Nuevo interfaz de usuario, 2ª cámara, mejoras de eficiencia, soporte nativo a VoIP, soporte NFC, incorporación de la Open Accessory Library (comunicación y reconocimiento USB).
22/02/2011	Honeycomb	3.x	11-13	Específica para tabletas (poco utilizada). Introduce las ActionBar y los Fragments, así como un nuevo motor gráfico, soporte procesadores multinúcleo, soporte a transmisiones en tiempo real (RTP), sincronización multimedia desde SD.

Lanzamiento	Nombre Comercial	Versión	API	Características relevantes
19/10/2011	Ice scream sandwidch	4.0.x	14-15	Unifica las versiones 2.x y 3.x y vale para todo tipo de dispositivo, trabajo sin botones físicos, mejoras en el reconocimiento de voz, reconocimiento facial, gestor de tráfico por internet.
27/06/2012	Jellybean	4.1	16	IGU más rápido y fluido, notificaciones expandibles y personalizables, widgets de tamaño variable, dictado por voz offline, Google Search, Google Now, mejoras de cifrado de apps y actualizaciones parciales.
29/10/2012	Jellybean (Gummy bear)	4.2	17	Varias cuentas de usuario en el mismo dispositivo (tabletas), widgets en la pantalla de bloqueo, nuevo teclado predictivo deslizante al estilo Swype, posibilidad de conectar los dispositivos a TV HD por WIFI (miracast).
24/07/2013	Jellybean (Gominola)	4.3	18	Soporte bluetooth 4.0, mejora en la escritura, modo de perfiles con acceso restringido, cambio de usuarios más rápido, localización por WIFI en segundo plano, mejoras de seguridad, OpenGL ES 3.0.
31/10/2013	KitKat	4.4	19	Mejoras de rendimiento (menor consumo de memoria y batería), mejoras bluetooth, MAP, AVRCP 1.3, integración con almacenamiento en la nube, impresión inalámbrica.

Tabla 1 : Versiones y características de Android **(Ruiz & Andrés, 2014)**

Cuando se crea un nuevo proyecto con Android Studio, pide entre otras cosas, la versión mínima del SDK (Minimum SDK) que soportará la aplicación. Esto quiere decir que, cualquier dispositivo con una versión inferior de Android, no podrá instalarse la app y obviamente no podrá utilizarla. También hay que tener en cuenta que no tendremos disponible características ofrecida en versiones posteriores (aunque este inconveniente se puede solucionar con bibliotecas de compatibilidad) .

Se ha seleccionado como versión mínima Android 2.3.3 – 2.3.7 (API 10) , ya que proporciona todas las características necesarias para el desarrollo de la aplicación y además da cobertura a una gran cantidad de usuarios de dispositivos Android.

Se puede observar en la siguiente imagen que seleccionando la API 10 se proporciona una cobertura aproximada del 99,2% de los dispositivos con los servicios Google Play Store activados.

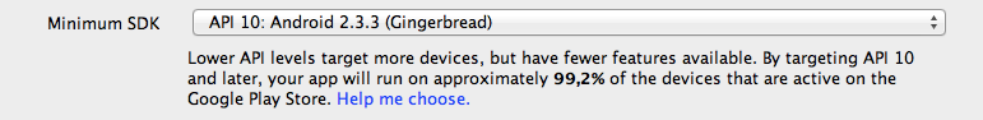

Ilustración 3: SDK mínimo, configuración de proyecto con Android Studio

Además, en cualquier versión del SDK se pueden utilizar bibliotecas de compatibilidad. Estas incorporan características importantes que no están soportadas en algunas versiones, como pueden ser, Fragments , Action Bar y Navigation Drawer que se utilizarán en la app.

2.3. Aplicaciones con funcionalidades similares

Este apartado presenta las aplicaciones que serán analizadas en los siguientes apartados. Se muestra una captura de pantalla de cada aplicación y una breve explicación de cada una con información extraída de Google Play.

Casa Domótica con Arduino Mega

Es una app que permite un control domótico de una vivienda con tan solo: Arduino Mega, Arduino Ethernet Shield y Real Time Clock Module RTC DS1307. Proporcionan el código del programa base de Arduino para poder modificarlo. La aplicación permite: control de iluminación con control de potencia, control de persianas o toldos, control de climatización, control de riego automático, control para puertas, etc.

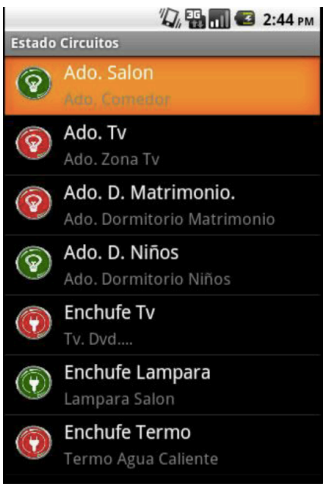

Ilustración 4: Captura de Casa Domótica con Arduino Mega

Winkhel Building Automation

Winkhel Building Automation es una aplicación que permite el control domótico de instalaciones (viviendas, oficinas, hoteles…). La aplicación hace uso del protocolo MODBUS TCP para comunicarse con una instalación basada en la tecnología Winkhel: una serie de dispositivos que heredan las características de Arduino y que, por tanto, son compatibles con él .

Ilustración 5: Captura de Winkhel Building Automation

e-Domótica

e-Domótica es una app que permite controlar el sistema e-Domótica (sistema propio) de forma remota desde cualquier lugar. e-Domótica se centra en el interior de la casa. Permite: identificarse en el sistema de forma fácil, poner en marcha o parar escenarios, poner en marcha o parar dispositivos, ver el estado de los dispositivos, ver la señal de video de las cámaras de seguridad y controlar la alarma.

Ilustración 6: Captura de e-Domótica

Control Domótica Raspberry

Control Domótica Raspberry es una app para controlar el sistema de domótica propio de la empresa. Permite controlar el sistema mediante internet o bluetooth. El sistema utiliza como servidor una RaspberryPI con conexión a internet.

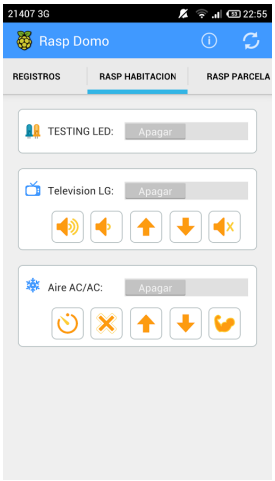

Ilustración 7: Captura de Control Domótica Raspberry

KASAMAGIKA domótica X-10

KASAMAGIKA domótica X-10 permite configurar y administrar una central domótica basada en Arduino Mega. La unidad de control es un pequeño servidor web conectado a un router. Los actuadores utilizan el protocolo X10 y TCP / IP. Un dato interesantes es que la app puede ser controlada por voz.

Ilustración 8: Captura de KASAMAGIKA domótica X-10

Arduino aire acondicionado

Arduino aire acondicionado es una app para el control de temperatura, humedad, punto de rocío e iluminación ambiental. Está pensada para recintos con aire acondicionado, calefactores y riego.

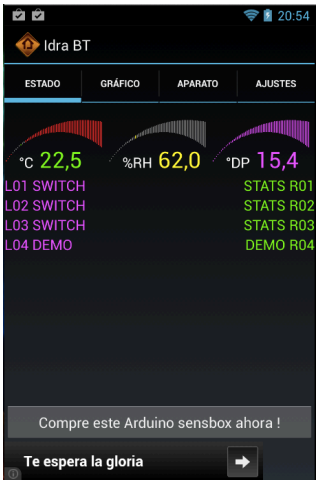

Ilustración 9: Captura de Arduino aire acondicionado

ImperiHome

ImperiHome permite controlar sistemas de automatización para hogar . No hay necesidad de un servidor central o instalación dedicada. La aplicación se conecta directamente a los sistemas que utilizan las API y permite interactuar de forma remota con todos sus dispositivos.

Ilustración 10: Captura de ImperiHome

Control4® MyHome

Control4® MyHome es una app que convierte el Smartphone o Tablet en un mando a distancia para controlar el sistema Control4 (sistema propio de la empresa). El sistema Control4 permite: reproducción de audio, reproducción de video, control de aire acondicionado/calefacción, visualización de cámaras IP, etc.

Ilustración 11: Captura de Control4MyHome

Vimar By-phone

Vimar By-phone permite gestionar del sistema domótico By-me y ClimaPhone directamente desde dispositivos Android. Se pueden controlar luces, persianas, temperatura, escenarios y el sistema de alarma del hogar .

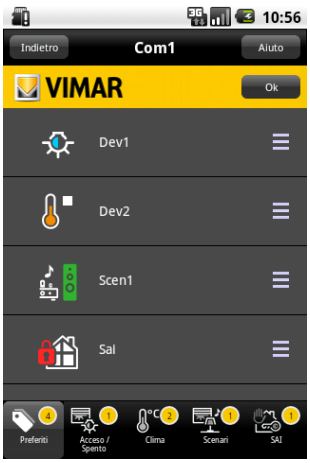

Ilustración 12: Captura de Vimar By-phone

2.4. Análisis cuantitativo

En este apartado se realiza un análisis cuantitativo de aplicaciones similares, basándose en datos recogidos en Google Play Store , como son la fecha de la última actualización, la versión mínima de Android requerida, número de instalaciones y puntuación de la app:

Aplicación	Último update	Versión mínima	Nº Instal.	Puntuación
Casa Domótica con Arduino Mega	29/11/2013	Android 1.6	5.000 - 10.000	4,7/5 de un total de 58 opiniones
Winkhel Building Automation	19/11/2012	Android 2.2	1000-5000	4,5/5 de un total de 23 opiniones
e-Domótica	13/06/2013	Android 3.2	500-1000	3,8/5
Control Domótica Raspberry	11/02/2014	Android 4.1	10-50	5/5 de un total de 1 opinión
KASAMAGIKA domótica X-10	08/05/2013	Android 2.2	100-500	-
Arduino aire acondicionado	06/01/2014	Android 2.1	1000-5000	4,2/5 de un total de 6 opiniones
ImperiHome	11/02/2014	Android 2.3.3	10.000 – 50.000	4,4/5 de un total de 256 opiniones
Control4® MyHome	08/10/2013	Android 2.2	50.000 – 100.000	4,5/5 de un total de 378 opiniones
Vimar By-phone	28/08/2013	Android 2.1	5.000 – 10.000	3,8/5 de un total de 37 opiniones

Tabla 2: Análisis cuantitativo

2.5. Análisis cualitativo

Por otro lado se han probado las aplicaciones anteriormente mencionadas para realizar un análisis que va más de allá de simples datos. Para hacer esto se ha utilizado Genymotion.

Genymotion es un emulador de Android que aprovecha la arquitectura x86 para ejecutar de forma fluida y rápida distintos dispositivos Android. Olvidando la lentitud del emulador nativo de Android, podemos ejecutar todo tipo de aplicaciones y juegos en nuestro Windows, Mac o Linux (Rodríguez, 2014).

En la siguiente tabla se muestra en una escala del 1-10, diferentes aspectos relevantes en la experiencia de usuario que han sido probados y puntuados:

Aplicación	Facilidad instalación	Facilidad uso	Facilidad Configuración	Interfaz de usuarios
Casa Domótica con Arduino Mega	10	7	5	5
Winkhel Building Automation	10	6	6	3
e-Domótica	10	6	8	6
Control Domótica Raspberry	10	8	6	5
KASAMAGIKA domótica X-10	10	5	3	3
Arduino aire acondicionado	10	5	3	5
ImperiHome	10	6	3	6
Control4® MyHome	10	6	6	8
Vimar By-phone	10	6	4	5
Media	10	6,1	4,9	5,1

Tabla 3: Análisis cualitativo

2.6. Análisis estratégico

Por último, se ha realizado un análisis estratégico centrado en información relacionada con el modelo de negocio, como es, la inclusión de publicidad dentro de la app, el precio y si hay versión de pago.

Aplicación	Publicidad app	Precio	Versión de pago
Casa Domótica con Arduino Mega	Si	Gratis	Si
Winkhel Building Automation	No	Gratis	No
e-Domótica	No	Gratis	No
Control Domótica Raspberry	No	Gratis	No
KASAMAGIKA domótica X-10	No	Gratis	No
Arduino aire acondicionado	Si	Gratis	Si
ImperiHome	No	Gratis	Si
Control4® MyHome	No	Gratis	No
Vimar By-phone	No	Gratis	No

Tabla 4: Análisis estratégico

2.7. Conclusiones

A partir de los análisis anteriormente detallados, se ha llegado a una serie de conclusiones que se han tenido en cuenta a la hora de desarrollar la app.

Todas la aplicaciones son gratuitas. Este dato es significativo pero no preocupante, puesto que el modelo de negocio no va orientado a sacar beneficio económico con la app en si. La aplicación se entrega junto al paquete del sistema de control de entorno exterior (ArduEntorno), con lo que se podrá bajar gratuitamente de la Google Play Store. Además, hay que tener en cuenta que solo se podrá utilizar la aplicación si se ha comprado dicho paquete.

Por esta misma razón, no se incluirá publicidad dentro de la app y tampoco habrán versiones de pago.

Otro dato interesante es el echo de que 8 de las 10 aplicaciones analizadas son compatibles con la API 10 o inferior, con lo que llegan a una gran cantidad de usuarios. Esto está resuelto al haber seleccionado como Minimum SDK la API 10.

Por último, es importante resaltar que los dos peores datos de usabilidad de las apps analizadas son la facilidad de configuración (4,9) y la interfaz de usuario (5,1). Son dos aspectos importantes a tener en cuenta que serán cubiertos por la app para diferenciarla y destacarla del resto de apps que se han analizado. Esto se conseguirá con una interfaz fácil de utilizar, e intuitiva.

3. Especificación de requisitos

3.1. Introducción

El apartado 3 describe la especificación de requisitos (ERS) de la app para la monitorización y gestión del sistema de control inteligente de entornos exteriores. Este apartado sigue la estructura definida y las prácticas recomendadas en el estándar IEEE 830-1998.

3.1.1. Propósito

El presente apartado tiene como propósito definir los requisitos funcionales y no funcionales de la app para el sistema de control inteligente de entornos exteriores. Está dirigido al equipo de desarrollo de la app y servirá como base para la construcción de la misma. Además permite llegar a un acuerdo con el cliente para definir que es lo que tiene que hacer la aplicación.

3.1.2. Ámbito del sistema

El objetivo principal que se persigue con el desarrollo de la aplicación es ofrecer al usuario la posibilidad de monitorizar y gestionar el estado del sistema de control inteligente de entornos exteriores desde cualquier lugar de forma fácil y con seguridad.

Desde la aplicación se podrán listar todas las áreas del entorno. Una vez seleccionada un área se podrán realizar una serie de operaciones sobre ella y ver el estado de los elementos de control (sensores y actuadores) que pertenecen a de dicha área.

3.1.3. Acrónimos

Nombre	Descripción
ERS	Especificación de Requisitos Software
RFXX	Se utilizará la siguiente nomenclatura en la numeración de los requisitos funcionales: R: Requisito F: Funcional XX: numeración ascendente
RNFXX	Se utilizará la siguiente nomenclatura en la numeración de los requisitos NO funcionales: R: Requisito N: No F: Funcional XX: numeración ascendente

Tabla 5: Acrónimos

3.1.4. Definiciones

Nombre	Descripción
Área	Es un espacio acotado dentro de un entorno. Dentro de un jardín se pueden tener por ejemplo: el área de entrada, el área de plantación, el área de ocio, etc.
Elemento de control	Son los sensores o actuadores que conforman el sistema.
Sensor	Dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc. (Wikimedia, 2014)
Actuador	Dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, una válvula. (Wikimedia, 2014)
ArduEntorno	Proyecto en el que está integrado la app. Este proyecto tiene como finalidad crear un sistema de control inteligente para entornos exteriores
iEnvironment	Aplicación para dispositivos móviles que permitirá a los usuarios controlar ArduEntorno desde teléfonos móviles y tablets Android.
Action Bar	El Action Bar es un componente de interfaz para Android donde se muestra el icono de la aplicación, indica al usuario donde se encuentra y proporciona una serie de acciones. Se muestra en forma de barra y se sitúa en la parte superior de la pantalla.
Navigation Drawer	Es otro componente de interfaz para Android que se presenta en forma de menú lateral y se sitúa en el lado izquierdo. Este menú se puede mostrar/ocultar pulsando sobre el icono de la aplicación que podemos encontrar en el Action Bar.
JPA	API para la persistencia de datos desarrollada para aplicaciones Java.

Tabla 6: Definiciones

3.1.5. Referencias

IEEE 830-1998: Recommended Practice for Software Requirements Specifications

3.1.6. Visión general de la especificación de requisitos

El punto de especificación de requisitos consta de 3 secciones:

- La primera sección (sección actual) realiza una breve introducción al apartado de especificación de requisitos.
- La segunda sección describe todos aquellos factores que afectan al producto y a sus requisitos. No se describen los requisitos, sino su contexto. Además proporciona una visión muy general del sistema.
- Por último, la tercera sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos.

3.2. Descripción general

Antes de detallar los requisitos del sistema, es importante hacer una descripción del sistema de alto nivel y analizar aquellos factores que afectan al producto y a sus requisitos. Esta sección incluye: perspectiva del producto, funciones del producto, características de los usuarios, restricciones y suposiciones.

3.2.1. Perspectiva del producto

iEnvironment forma parte de un proyecto de mayor envergadura llamado ArduEntorno. ArduEntorno es un sistema de control inteligente de entornos exteriores como por ejemplo, jardines, espacios lúdicos, plantaciones, etc.

iEnvironment actuará como interfaz de comunicación y gestión entre ArduEntorno y los usuarios en forma de app que permita a los usuarios leer el estado y controlar ArduEntorno desde dispositivos móviles Android.

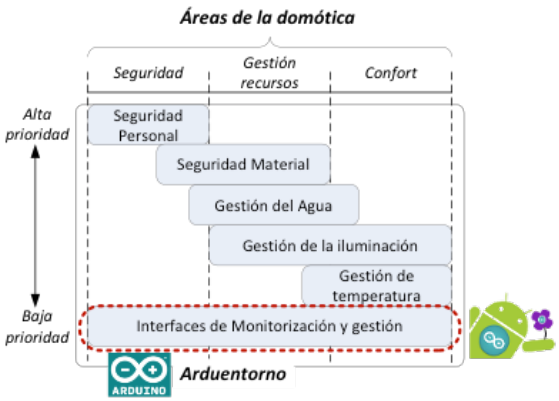

Ilustración 13: Módulos de ArduEntorno

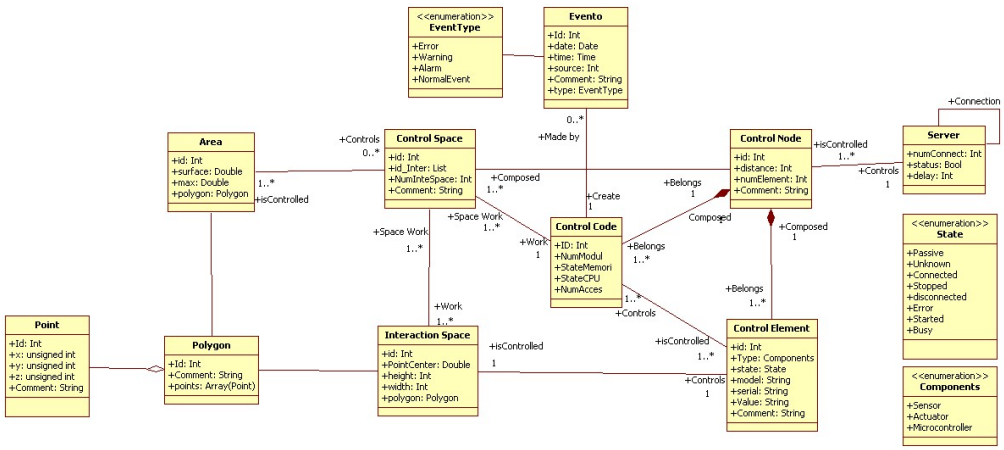

Ilustración 14: Ontología de control de espacios abiertos

3.2.2. Funciones del producto

En términos generales el producto tendrá que proporcionar a usuarios identificados la posibilidad de leer el estado del ArduEntorno y realizar operaciones sobre este. También facilitará la configuración de la aplicación "IP del servidor" y acceder al "About".

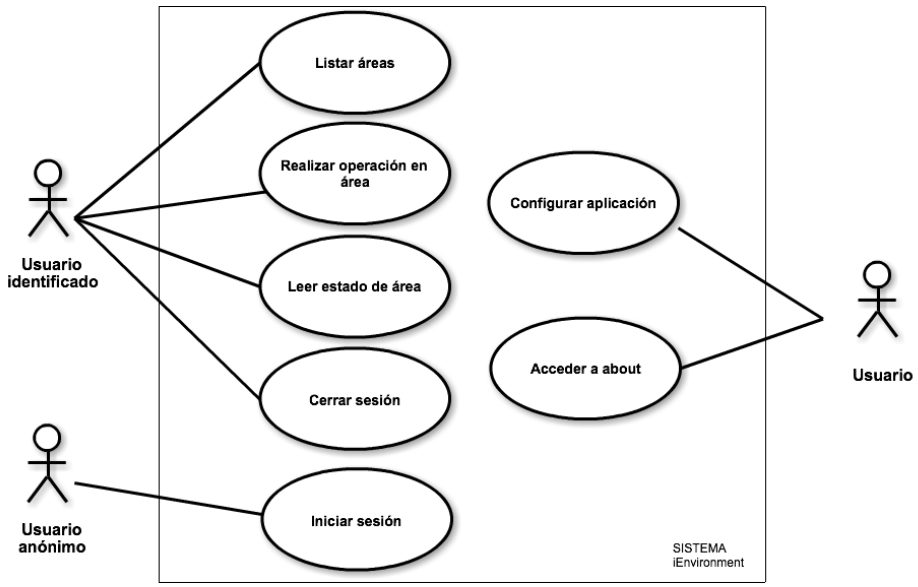

Ilustración 15: Diagrama de casos de uso

3.2.3. Características de los usuarios

Los potenciales usuarios de iEnvironment no tienen porqué tener conocimientos avanzados de informática. Es por esto que la aplicación tendrá que ser fácil de usar e intuitiva, haciendo sencillo el aprendizaje de utilización. Por otro lado, también tendrá que ser fácil de configurar.

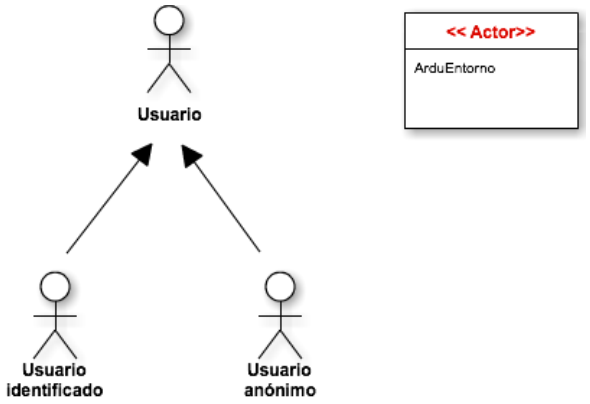

Ilustración 16: Diagrama de actores del sistema

Tipo de usuario	Usuario
Formación	No especificada
Actividades	Configurar la aplicación y acceder al About.

Tipo de usuario	Usuario identificado
Formación	No especificada
Actividades	Realizar operaciones sobre ArduEntorno y monitorizar el estado de los sensores y actuadores. También podrá cerrar la sesión iniciada.

Tipo de usuario	Usuario anónimo
Formación	No especificada
Actividades	Iniciar sesión.

3.2.4. Restricciones

- La aplicación será compatible con dispositivos Android con API 10 o superior.
- La base de datos tendrá que estar implementada en MySQL.
- Los servicios web seguirán la arquitectura REST y estará implementado en java.
- El servidor web donde se desplieguen los servicios web tendrá que ser Tomcat 7.

3.2.5. Suposiciones y dependencias

- Se asume que el sistema de control inteligente de entornos exteriores ArduEntorno dispondrá de un servidor.
- El servidor tendrá una IP pública y esta será fija.
- Este Servidor tendrá instalado MySQL y Apache Tomcat.
- El puerto de MySQL será el 3306 y el de Tomcat el 8080.
- Estos puertos estarán accesibles desde IPs externas al sistema

3.3. Requisitos específicos

Esta sección detalla los requisitos funcionales y no funcionales con suficiente nivel de detalle para que el equipo de desarrollo pueda cubrir y comprobar todas las necesidades expuestas por el cliente. Todos los requisitos tendrán que ser cubiertos por el sistema.

3.3.1. Interfaces externas

- Interfaz de usuario: La interfaz de la aplicación tendrá que ser intuitiva y fácil de usar. Mostrará una barra superior "Action Bar" con acceso a la sesión de usuario, la configuración de la aplicación y el "Acera de" de la misma. Además, la aplicación tendrá un menú lateral "Navigation Drawer" donde mostrará las diferentes áreas del entorno.

- Interfaz hardware: El usuario podrá interactuar con la aplicación a través de la pantalla del dispositivo móvil. Además, estos dispositivos podrán incluir teclado físico el cual también podrá ser usado para la introducción de texto.

- Interfaz software: Los dispositivos que no tengan teclado físico utilizaran el teclado integrado en la interfaz de Android.

- Interfaz de comunicaciones: La aplicación se comunicará con los servicios web a través del protocolo HTTP. Se podrán hacer las siguientes operaciones: POST, GET, PUT y DELETE. Además los servicios web se comunicarán con la base de datos a través de JPA.

3.3.2. Requisitos funcionales

Identificador	RF01
Nombre	Iniciar sesión
Descripción	El usuario podrá iniciar sesión si introduce nombre de usuario y contraseña y pulsa sobre el botón de inicio de sesión.
Entradas	Nombre de usuario y contraseña
Salidas	Si el usuario y/o la contraseña son incorrectos se mostrará un mensaje de error y no se permitirá iniciar sesión. En caso contrario se mostrará una notificación confirmando el inicio de sesión.

Identificador	RF02
Nombre	Cerrar sesión
Descripción	Cuando el usuario acceda a la ventana de sesión de usuario y la sesión haya sido iniciada con anterioridad, el usuario podrá cerrar sesión. El usuario solo tendrá que pulsar sobre el botón de cerrar sesión.
Entradas	Ninguna
Salidas	Si el proceso se ha realizado correctamente se mostrará un mensaje de confirmación de la operación. En caso contrario se mostrará un mensaje de error.

Identificador	RF03
Nombre	Mostrar IP servidor
Descripción	Cuando el usuario acceda a la ventana de configuración de la aplicación, este podrá visualizar la IP del servidor almacenada en memoria.
Entradas	Ninguna
Salidas	Se mostrará la IP almacenada en memoria. En caso de no haber podido recuperar ninguna IP, se mostrará por defecto : 000.000.000.000

Identificador	RF04
Nombre	Guardar IP servidor
Descripción	Cuando el usuario acceda a la ventana de configuración de la aplicación, este podrá modificar la IP del servidor de ArduEntorno. Para modificar la IP solo tendrá que introducir los dígitos de la dirección IP y pulsar en el botón de guardar.
Entradas	Los cuatro campos de la dirección IP: XXX . XXX . XXX . XXX
Salidas	Si alguno de los cuatro campos de la dirección IP son nulos, se mostrará un mensaje de error y no se modificará la IP. En caso contrario, se mostrará un mensaje de operación realizada correctamente.

Identificador	RF05
Nombre	About
Descripción	El usuario podrá acceder a la ventana de información de la aplicación. Esta mostrará un texto explicativo de la aplicación y permitirá enviar un correo electrónico al desarrollador de la misma.
Entradas	Ninguna
Salidas	Si se pulsa el correo electrónico del desarrollador, la aplicación abrirá el gestor de correo de Android. En caso de tener varios gestores de correo, se mostrará un menú para seleccionar el gestor que se desea usar.

Identificador	RF06
Nombre	Listar áreas
Descripción	El usuario, si ha iniciado sesión, podrá listar las áreas del entorno al pulsar en el icono de la aplicación ubicado en el Action Bar. Con esto se mostrará todas las áreas en el menú lateral del Navigation Drawer
Entradas	Ninguna
Salidas	Al pulsar el icono de la aplicación se mostrará/ocultará el menú lateral del Navigation Drawer con el listado de todas las áreas.

Identificador	RF07
Nombre	Listar operaciones de área
Descripción	Se podrá acceder al listado de operaciones disponibles de un área cuando se pulse sobre un área en el listado del Navigation Drawer. Además se mostrará el estado de cada operación (ON / OFF)
Entradas	Ninguna
Salidas	Se mostrará un listado con todas las operaciones.

Identificador	RF08
Nombre	Cambiar estado de operación
Descripción	Cuando se pulse sobre una operación del listado de operaciones disponibles de un área, se podrá activar o desactivar dependiendo de su estado actual.
Entradas	Ninguna
Salidas	Se mostrará un mensaje indicando si la operación se ha podido realizar o si ha habido algún error. Se cambiará el estado de la operación.

Identificador	RF09
Nombre	Leer estado de área
Descripción	La aplicación mostrará un listado con los sensores y actuadores del área y el estado de estos. Se podrá acceder a esta funcionalidad pulsando sobre un área en el listado de del Navigation Drawer y accediendo a la pestaña de estado.
Entradas	Ninguna
Salidas	Se mostrará el listado de todos los sensores y actuadores que están integrados dentro del área junto con el estado de estos.

3.3.3. Requisitos NO funcionales

Identificador	RNF01
Tipo	Seguridad
Nombre	Control acceso
Descripción	Para poder gestionar ArduEntorno desde la app, será necesario iniciar sesión correctamente en el sistema.

Identificador	RNF02
Tipo	Seguridad
Nombre	Usuario, contraseña y cadena de conexión
Descripción	Tanto el nombre de usuario, contraseña y cadena de conexión que maneja el sistema se enviarán cifrados.

Identificador	RNF03
Tipo	Fiabilidad
Nombre	Control de excepciones
Descripción	Tanto la app como los servicios web tendrán que controlar las excepciones, evitando el cierre inesperado de la app.

4. Diseño del sistema

4.1. Introducción

Este apartado se centra en el diseño tanto formal como conceptual del sistema. Dentro del diseño conceptual se ha incluido el diseño de la arquitectura del sistema. Por otro lado, el diseño formal contiene los prototipos de la app, diagramas de flujo y el modelo de la base de datos. Por último se ha incluido el icono de la app.

4.2. Diseño conceptual del sistema

4.2.1. Arquitectura

Se ha optado por una arquitectura cliente-servidor. El servidor tendrá alojado los servicios web y la base de datos. La app solo se podrá comunicar con el sistema a través de los servicios web. Los Arduinos estarán conectados a la misma red de área local que el servidor y se comunicarán directamente con la base de datos.

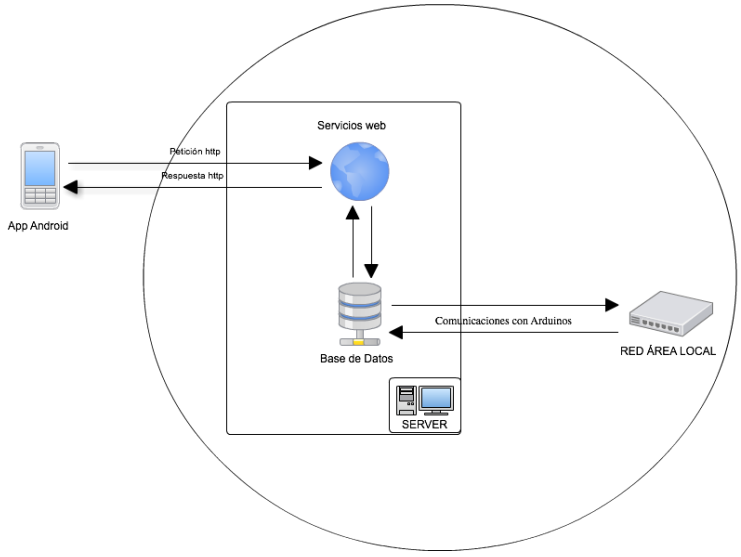

Ilustración 17: Diseño de la arquitectura del sistema

4.3. Diseño formal del sistema

4.3.1. Prototipos

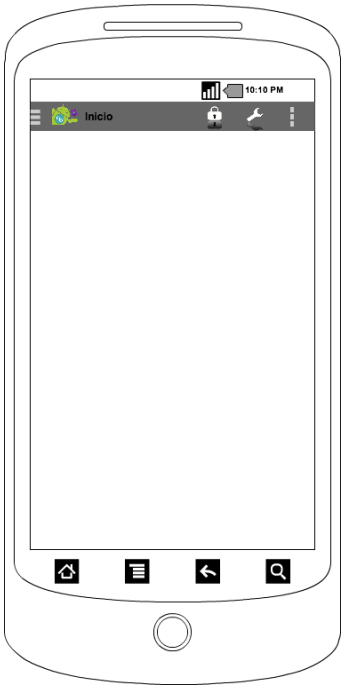

Ilustración 18: Prototipo de pantalla de inicio

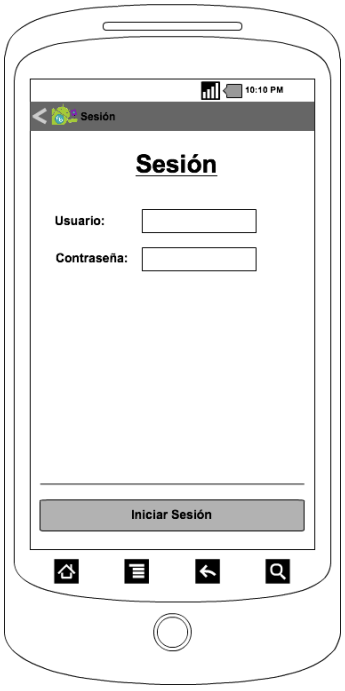

Ilustración 19: Prototipo de pantalla de sesión de usuario

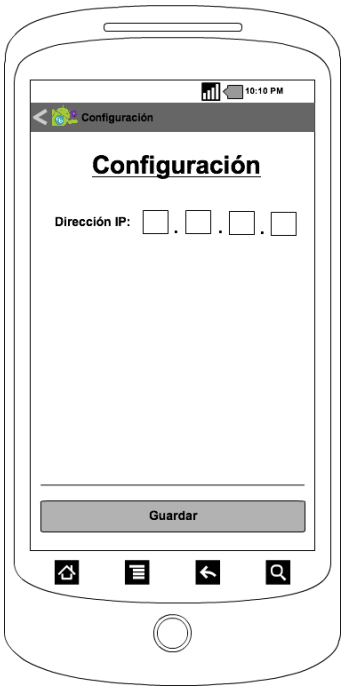

Ilustración 20: Prototipo de pantalla de configuración

Ilustración 21: Prototipo de pantalla “Acerca de”

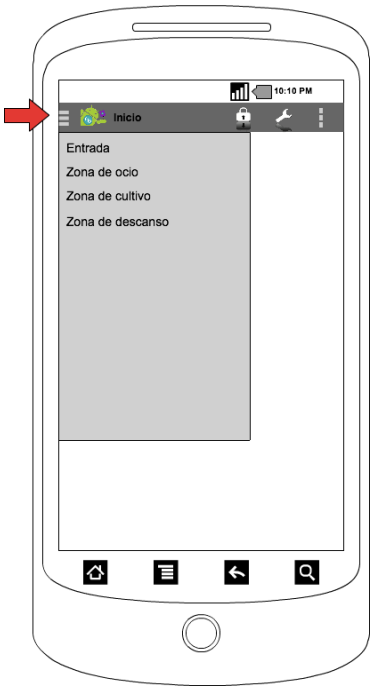

Ilustración 22: Prototipo de listado de áreas

Ilustración 23: Prototipo de pantalla de gestión de área

4.3.2. Diagramas de flujo

4.3.2.1. Mostrar IP del servidor

Para mostrar la IP del servidor, primero se comprueba si ha sido almacenada anteriormente en memoria interna de la app. Si la IP se ha podido recuperar, se muestra en la interfaz. En caso contrario se muestra la IP por defecto "000.000.000.000".

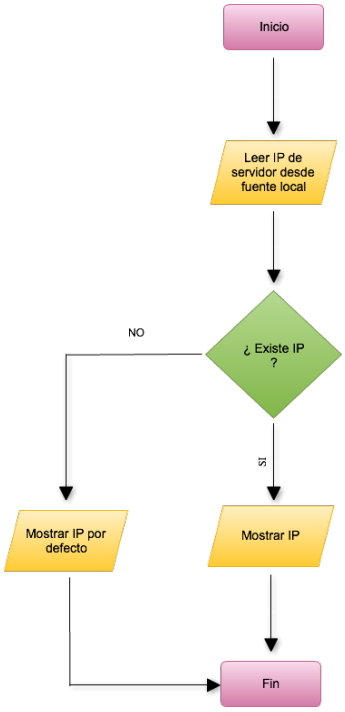

Ilustración 24: Diagrama de flujo - Mostrar IP de servidor

4.3.2.2. Guardar IP del servidor

Para guardar la IP del servidor, primero se comprueba que la IP introducida por el usuario en la interfaz de la app es correcta. En caso de ser correcta, se almacena en memoria interna de la app y se muestra un mensaje de confirmación. En caso contrario, se muestra un mensaje de error.

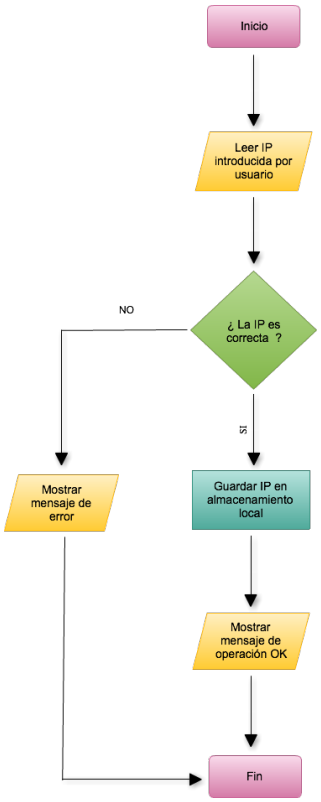

Ilustración 25: Diagrama de flujo - Guardar IP de servidor

4.3.2.3. Iniciar sesión

Para iniciar sesión, primero se comprueba si hay conexión a internet. En caso de haber conexión a internet, se lee la IP del servidor almacenada en memoria local de la app. Si existe la IP del servidor, se envía una petición al servicio web para que compruebe si el usuario y la contraseña introducidos son correctos. En caso de ser correctos, el servicio devuelve la cadena de conexión que será almacenada en memoria local junto al estado de la sesión (conectado). En caso de producirse algún error, se muestra un mensaje informando al usuario.

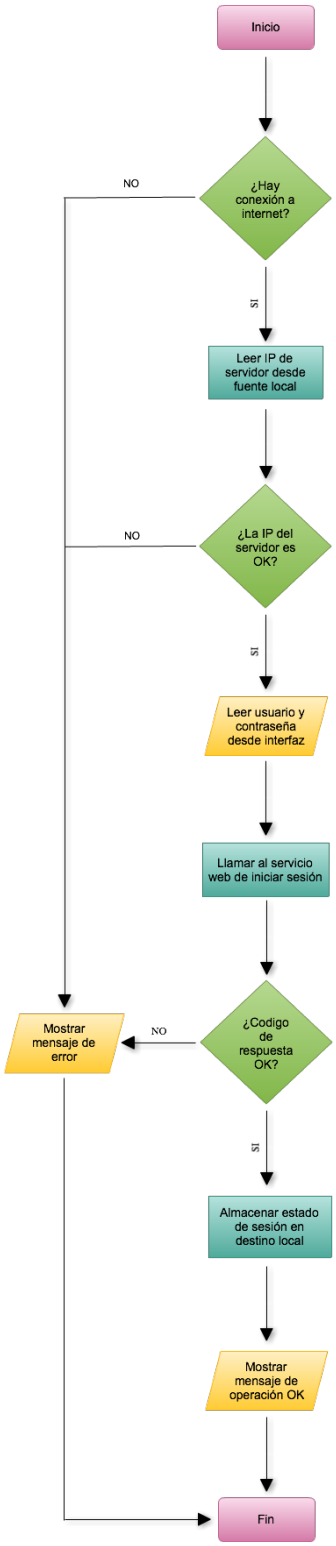

Ilustración 26: Diagrama de flujo - Iniciar sesión

4.3.2.4. Cerrar sesión

Para cerrar la sesión de usuario, se modifica el estado de la sesión y se almacena en memoria interna de la app. Si todo ha ido bien, se muestra un mensaje de confirmación. En caso contrario, se muestra un mensaje de error.

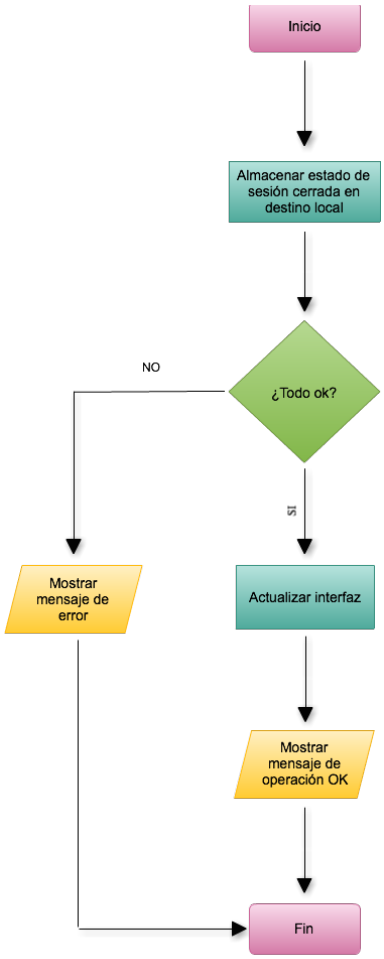

Ilustración 27: Diagrama de flujo - Cerrar sesión

4.3.2.5. Listar áreas

Para listar las áreas del entorno, primero se comprueba si hay conexión a internet. Si hay conexión de internet, se realiza una petición al servicio web para obtener las áreas del entorno. Si el código de respuesta es correcto, se muestran las áreas. Si se produce algún error, se muestra un mensaje para indicarlo.

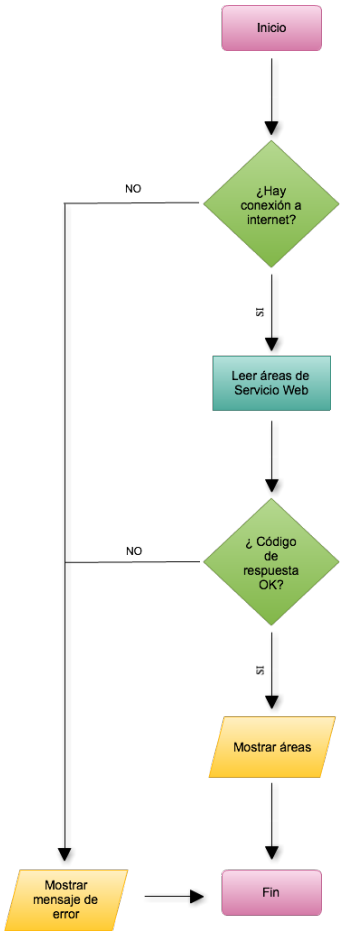

Ilustración 28: Diagrama de flujo - Listar áreas

4.3.2.6. Listar operaciones de área

Para listar las operaciones de un área, primero se comprueba si hay conexión a internet. En caso de haber conexión, se hace una petición al servicio web para obtener todas las operaciones posibles de un área. Si el código de la respuesta es correcto, se muestran las operaciones y el estado de las mismas. En caso de producirse algún error se muestra el mensaje correspondiente.

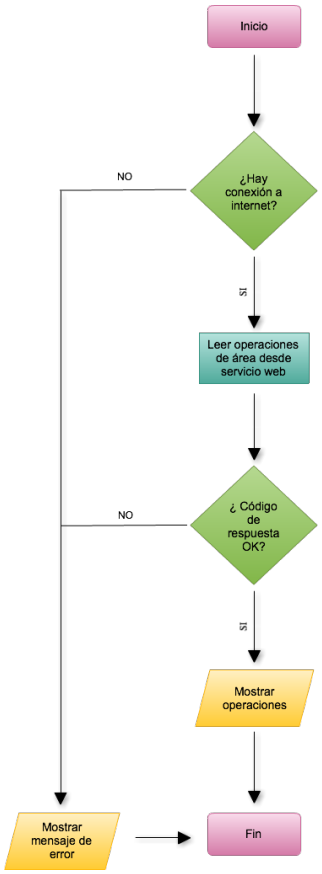

Ilustración 29: Diagrama de flujo - Listar operaciones de área

4.3.2.7. Cambiar estado de operación

Para cambiar el estado de una operación, primero se comprueba si hay conexión a internet. En caso de haber conexión de internet, se hace una petición al servicio para cambiar el estado de la operación seleccionada en el listado. Si el código de respuesta es correcto, se muestra un mensaje de confirmación y se actualiza la interfaz. En caso contrario, se muestra un mensaje de error.

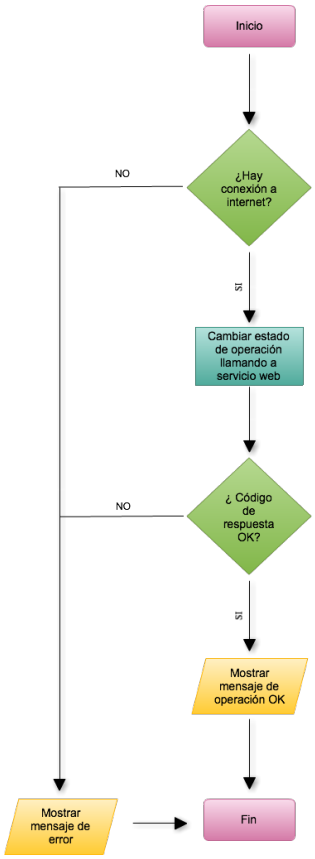

Ilustración 30: Diagrama de flujo - Cambiar estado de operación

4.3.2.8. Mostrar estado de área

Para mostrar el estado de un área, primero se comprueba si hay conexión a internet. En caso de haber conexión de internet, se realiza una petición al servicio al servicio web para obtener el estado de todos los elemento de control del área. Si el código de la respuesta es correcto, se muestra un listado con los elementos de control y los estados de los mismos. En caso contrario, se muestra un mensaje de error.

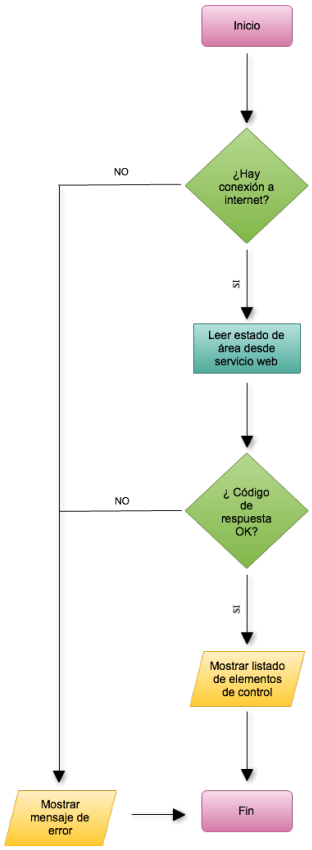

Ilustración 31: Diagrama de flujo - Mostrar estado de área

4.3.3. Base de datos

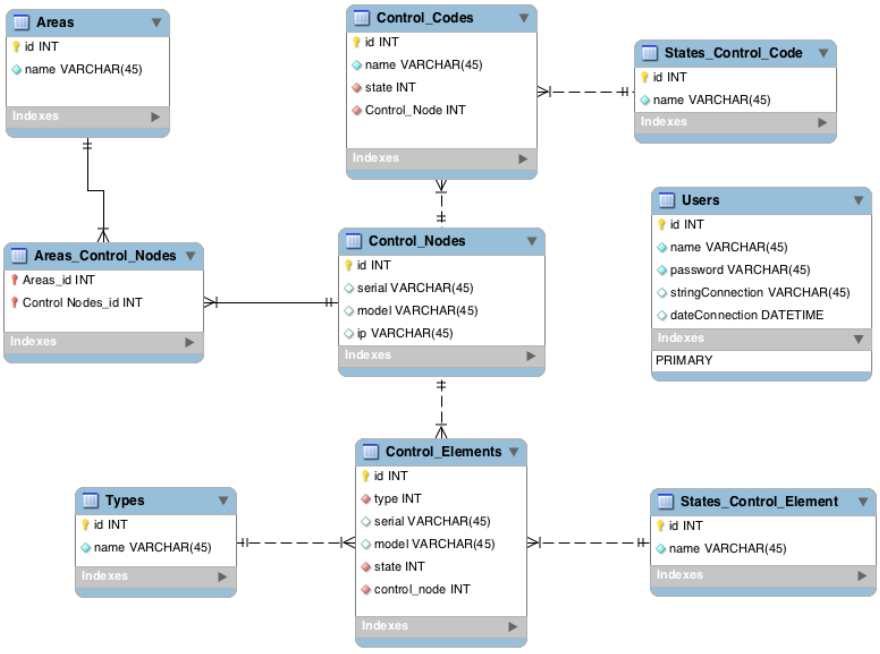

Ilustración 32: Modelo de la base de datos

4.4.Otros

4.4.1. Icono de la aplicación

Ilustración 33: Icono de la app

5. Implementación

5.1. Introducción

Este apartado contiene información relativa a la implementación de la aplicación, los servicios web y la base de datos. Se indica que entornos de desarrollo se han utilizado, se muestran algunos ejemplos de código para entender la conexión entre las distintas partes del proyecto y se mencionan decisiones de implementación.

5.2. Aplicación Android

La aplicación se ha desarrollado con Android Studio, IDE expresamente enfocado al desarrollo de aplicaciones para Android. Fue lanzado en mayo de 2013 por Google, compañía que también ha desarrollado Android.

Para la gestión local de información que la aplicación necesita para su correcto funcionamiento, como puede ser información de configuración o el estado de la sesión, se ha utilizado Shared Preferences. Es una solución sencilla que proporciona Android pero suficiente para las necesidades de la aplicación. La información se guarda en forma de clave-valor.

```
SharedPreferences sharedPreferences =
context.getSharedPreferences("preferencias",
Activity.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("user",user);
editor.commit();
```

Código 1: Ejemplo de guardado con Shared Preferences

```
SharedPreferences sharedPreferences =
context.getSharedPreferences("preferencias",
Activity.MODE_PRIVATE);
String user = sharedPreferences.getString("user",null);
```

Código 2: Ejemplo de lectura con Shared Preferences

Por otro lado, para las comunicaciones con los servicios web se ha utilizado AsyncTask y conexiones HTTP.

AsyncTask proporciona un mecanismo para ejecutar tareas en segundo plano. De esta manera, estas tareas se ejecutan en hilos diferentes al hilo que controla la GUI de la aplicación. Es interesante a la hora de ejecutar tareas costosas sin que afecten al rendimiento y a la interfaz de la aplicación.

Los métodos onPreExecute y onPostExecute se ejecutan en el hilo principal y lo hacen antes o después de la ejecución de la tarea asíncrona. Al ejecutarse en el hilo principal pueden modificar la interfaz de la aplicación.

El método doInBackground ejecuta el código en segundo plano usando un hilo diferente al hilo de la GUI de la aplicación. Este método no puede modificar la interfaz de la aplicación.

```
private class RestLogin extends
AsyncTask<Void,Void,Integer>{
    @Override
    protected void onPreExecute() {…}
    @Override
    protected Integer doInBackground(Void... params) {…}
    @Override
    protected void onPostExecute(Integer aInteger) {…}
```

Código 3: Ejemplo de AsyncTask

Para realizar conexiones HTTP con servicios web, se ha utilizado la clase httpURLConnection. Esta clase permite el intercambio de información a través de la web. Podemos realizar operaciones GET, POST, PUT y DELETE.

```
URL url = new
URL(stringUrl+"?"+URLEncodedUtils.format(pairs,"utf8"));
httpURLConnection = (HttpURLConnection)
url.openConnection();
httpURLConnection.setRequestMethod("GET");
httpURLConnection.setRequestProperty("Accept","text/plain");
bufferedReader = new BufferedReader(new
InputStreamReader(httpURLConnection.getInputStream()));
String response = bufferedReader.readLine();
```

Código 4: Ejemplo de conexión http con httpURLConnection

5.3. Servicios web

Los servicios web se han desarrollado en Java con el entorno de desarrollo Eclipse. Se ha utilizado JAX-RS, API que facilita la implementación de servicios web RESTful en Java.

JAX-RS proporciona soporte en la creación de servicios web de acuerdo con el estilo arquitectónico Representational State Transfer (REST). JAX-RS usa anotaciones introducidas en Java para simplificar el desarrollo y despliegue de los clientes y puntos finales de los servicios web (Wikimedia, 2013) .

```
@Path("/session/")
public class Session {

   @GET
   @Path("hello")
   @Produces("text/plain")
   public String getIt() {
       return "Hi there!";
    }

}
```

Código 5: Ejemplo de JAX-RS

Por otro lado, se ha utilizado Maven para la gestión de las librerías del proyecto. Maven utiliza un fichero Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos (Wikimedia, 2014). A continuación se muestra parte de este fichero:

```
<dependencies>
       <dependency>
           <groupId>org.hibernate.javax.persistence</groupId>
           <artifactId>hibernate-jpa-2.1-api</artifactId>
           <version>1.0.0.Final</version>
       </dependency>
       …
 </dependencies>
```

Código 6: Ejemplo de fichero POM.xml

5.4. Persistencia

La base de datos se ha implementado en MySQL y para el diseño se ha utilizado MySQL Workbench. MySQL Workbench es una herramienta visual que permite crear y mantener fácilmente bases de datos en MySQL .

Una vez se ha creado el modelo de la base de datos con MySQL Workbench, permite genera el script de creación o modificación de la base de datos. El modelo de la base de datos se puede observar en el apartado 4.3.3 .

Para la capa de persistencia de los servicios web se ha utilizado JPA, API para Java que permite manejar datos relacionales de forma sencilla al utilizar las ventajas de la programación orientada a objetos. Permite mapear tablas de una base de datos con objetos Java, haciendo mucho más fácil el manejo de los datos.

A continuación se muestra un fragmento de la clase “User” que mapea la tabla de la base de datos “Users”:

```
@Entity
@Table(name="Users")
@NamedQuery(name="User.findAll", query="SELECT u FROM User u")
public class User implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="id", unique=true, nullable=false)
	private Integer id;
```

Código 7: Ejemplo de JPA

5.5. Servidor

El proyecto ArduEntorno al estar todavía en fases iniciales, no dispone de la arquitectura necesaria. Es por esto que se ha configurado un servidor en la nube para simular el servidor que posteriormente estará integrado en el sistema. Para ello se ha utilizado AWS (Amazon Web Services), servicios en la nube proporcionados por Amazon. Concretamente se ha utilizado Amazon EC2, plataforma que permite a los usuarios utilizar máquinas virtuales y que dispone de una gran variedad de sistemas operativos. Se ha optado por una micro instancia con Windows Server 2008 donde se ha instado MySQL (puerto 3306) y Tomcat (puerto 8080).

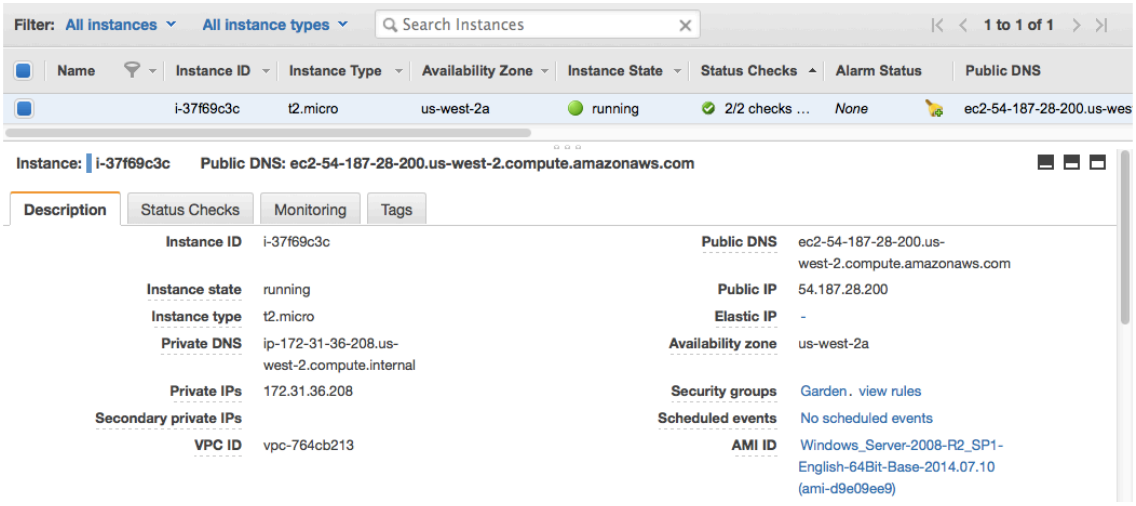

Ilustración 34: Amazon Web Services

5.6. Capturas de pantalla de la app

A continuación se muestran algunas capturas de la app funcionando. Las capturas mostradas se han obtenido ejecutando la app en Genymotion y corresponden a la home de la app, el inicio de sesión y el listado de las áreas recuperadas desde el servicio web.

Ilustración 35: Captura iEnvironment - Home

Ilustración 36: Captura iEnvironment - Inicio de sesión

Ilustración 37: Captura iEnvironment - Listado de áreas

6. Conclusiones

6.1. Introducción

En este apartado se analizan las dificultades que se han encontrado a la hora de llevar a cabo el proyecto, se recopilan las aportaciones que se han obtenido con la realización del mismo y se proponen una serie de ampliaciones futuras que se pueden desarrollar a partir del producto presentado.

6.2. Dificultades encontradas

A la hora de realizar el proyecto se han encontrado dificultades en cuanto a la inestabilidad del entorno de desarrollo (Android Studio) cuando se han instalado actualizaciones. Se ha optado por utilizar la última versión estable probada (0.4.2). Estos problemas han ocasionado perdida de trabajo implementado, por lo que se ha decidido que todos los cambios desarrollados tanto en la app, como en los servicio web, sean guardados en un sistema de control de versiones (GIT).

Por otro lado, se ha encontrado la problemática de no disponer de la infraestructura completa de ArduEntorno. Se ha solventado este inconveniente con la configuración y utilización de un servidor en la nube (Amazon Web Services)

6.3. Aportaciones obtenidas

En primer lugar, se ha analizado el entorno Android en general y las apps que hay en el mercado, vinculadas con la domótica. Se ha llegado a la conclusión de que la versión mínima de Android que tiene que soportar la aplicación es Android 2.3.3–2.3.7 (API 10). Además, se ha decidido que la aplicación sea gratuita (incluida en el paquete de ArduEntorno) y que se diferencie del resto por ser fácil de configurar y tener una interfaz intuitiva. Este análisis puede servir de punto de partida para la realización de aplicaciones Android en general y/o aplicaciones Android en el ámbito de la domótica.

Otra aportación interesante es la especificación de requisitos funcionales y no funcionales del sistema. Se ha decidido que la aplicación proporcione control de acceso para limitar los usuarios que puedan gestionar el sistema, que la información de sesión se envíe cifrada para aumentar la seguridad y se controlen todas las posibles excepciones para dar mayor consistencia a la app. Por otro lado, la aplicación permite configurar la dirección IP del servidor, iniciar/cerrar sesión y gestionar las distintas áreas del espacio controlado. Por lo tanto, el apartado 3 puede servir como ejemplo de especificación siguiendo el estándar IEEE 830-1998.

Además, se ha realizado un diseño conceptual y formal del sistema. Se ha decidido que la arquitectura sea cliente-servidor y que la aplicación se comunique con el sistema de control del entorno a través de servicios web. Además, se han diseñado las interfaces de la app y los diagramas de flujo de las funciones del sistema. Estas decisiones de diseño pueden dar ideas de cómo plantear el sistema desde el punto de vista del diseño.

Por último y no menos importante, se han decido detalles de implementación, como los lenguajes escogidos, los entornos de desarrollo utilizados y las APIs integradas. Para desarrollar la app se ha utilizado Android Studio. La persistencia de los datos locales de la app se gestionan con Shared Preferences y para las comunicaciones con los servicios web se utilizan conexiones HTTP y distintos hilos de ejecución. Los servicios web se han implementado en Java, utilizando Eclipse y la librería JAX-RS. Además, se ha usado JPA para la capa de persistencia de los servicios web y Maven para la gestión de librerías. La base de datos se ha implementado en MySQL y se ha creado y mantenido con MySQL Workbench. Este último punto puede resultar muy interesante ya que propone tecnologías actuales que facilitan la implementación de productos similares.

6.4.Ampliaciones futuras

El presente proyecto presenta diversos apartados que pueden ser ampliados o mejorados dependiendo de las necesidades del producto final:

- Ampliar la especificación de requisitos, aportando mayor funcionalidad al sistema.
- Mejorar la seguridad del la aplicación, incluyendo un sistema de cifrado más robusto.
- Diseñar una interfaz que se adapte mejor a tablets. Además, proporcionar la opción de utilizar la aplicación en modo horizontal. Actualmente solo se puede utilizar en modo vertical.

7. Bibliografía

Wikimedia. (28 de Julio de 2014). *Actuador*. Retrieved 18 de Agosto de 2014 from Wikipedia: http://es.wikipedia.org/wiki/Actuador

Wikimedia. (13 de octubre de 2013). *JAX-RS*. Retrieved 22 de agosto de 2014 from Wikipedia: http://es.wikipedia.org/wiki/JAX-RS

Wikimedia. (15 de Julio de 2014). *Maven*. Retrieved 27 de Agosto de 2014 from Wikipedia: http://es.wikipedia.org/wiki/Maven

Wikimedia. (10 de Junio de 2014). *Sensor*. Retrieved 18 de Agosto de 2014 from Wikipedia: http://es.wikipedia.org/wiki/Sensor

Android Open Source Project. (1 de Mayo de 2014). *Dashboard*. Retrieved 15 de Mayo de 2014 from Android Developers: http://developer.android.com/about/dashboards/index.html

Asociación Española de Domótica e Inmótica. (n.d.). *Qué es Domótica*. Retrieved 17 de Agosto de 2014 from CEDOM: http://www.cedom.es/sobre-domotica/que-es-domotica

Cornet, M. (4 de Diciembre de 2012). *Todas las versiones de Android, en un divertido gráfico de su evolución*. Retrieved 10 de Agosto de 2014 from ABCdesevilla (Mobility): http://sevilla.abc.es/mobility/noticia/android/noticias-android/todas-las-versiones-de-android-en-un-divertido-grafico-de-su-evolucion/

Ruiz, J. C., & Andrés, D. d. (2014). SDM - Tema 2: Fundamentos del desarrollo con Android - UPV. Valencia, Valencia, España.

Rodríguez, T. (17 de Marzo de 2014). *Genymotion, el emulador más rápido de Android*. Retrieved 11 de Agosto de 2014 from Xataka Android: http://www.xatakandroid.com/roms-android/genymotion-el-emulador-mas-rapido-de-android